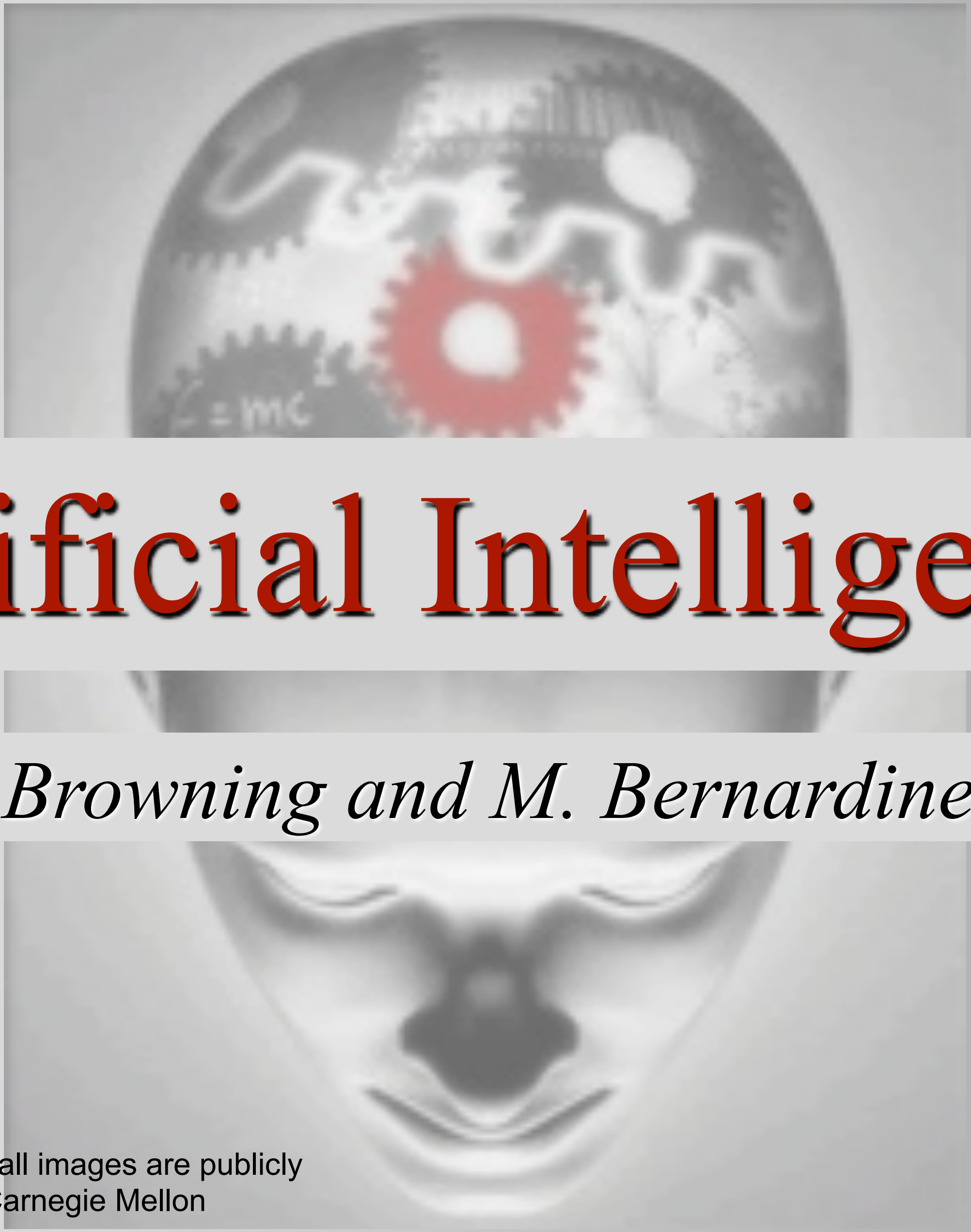
جامعة كارنيجي ميلون في قطر
Carnegie Mellon
QATAR CAMPUS



Artificial Intelligence

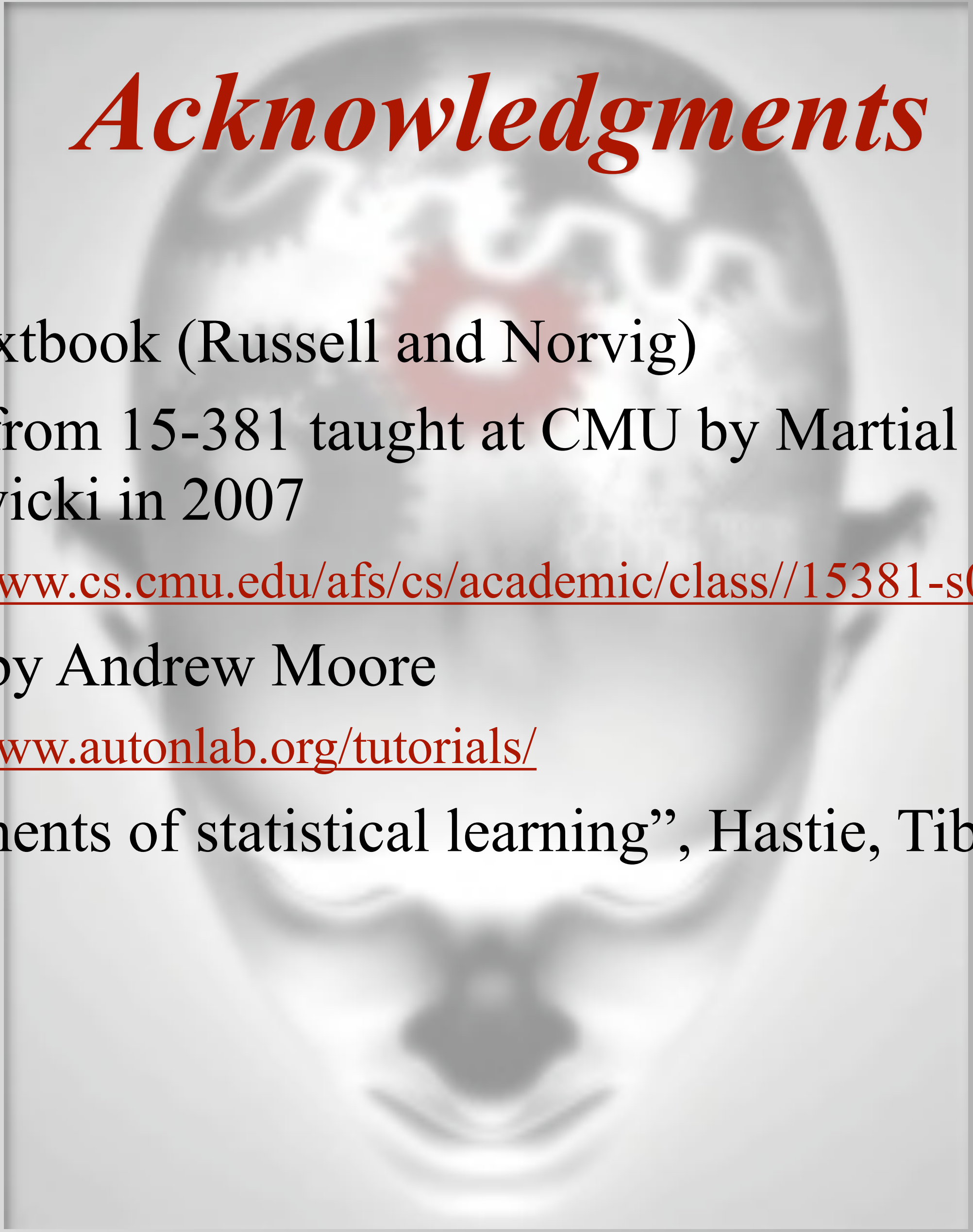
Brett Browning and M. Bernardine Dias

15-381

Spring 2008

Unless otherwise stated, all images are publicly available online or from Carnegie Mellon

http://www.actmedia.org/Email/TCRP/Images/Sept2006/ComputerBrain.gif



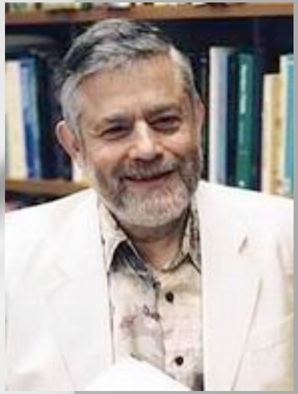
Acknowledgments

- Course textbook (Russell and Norvig)
- Lectures from 15-381 taught at CMU by Martial Hebert and Mike Lewicki in 2007
  - http://www.cs.cmu.edu/afs/cs/academic/class//15381-s07/www/
- Lectures by Andrew Moore
  - http://www.autonlab.org/tutorials/
- “The elements of statistical learning”, Hastie, Tibshirani, Friedman

Outline

- Recap: unsupervised learning
- Clustering
  - K-means clustering
  - Soft clustering with EM
  - Agglomerative clustering
  - Picking the number of clusters
  - Classification
- Dimensionality reduction
  - PCA
- Practical examples

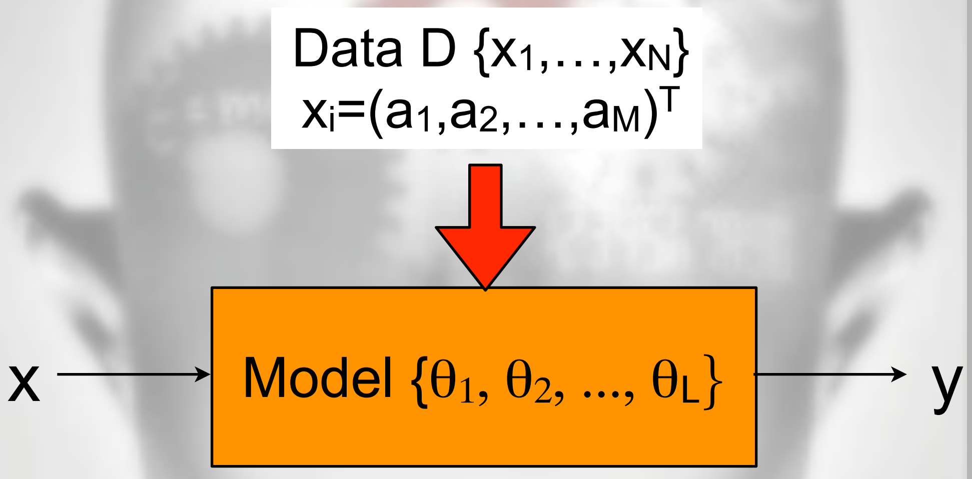
Manuel Blum



Bio from: http://awards.acm.org/citation.cfm?id=0451202&srt=all&aw=140&ao=AMTURING

Manuel Blum (born 26 April 1938) is a computer scientist who received the Turing Award in 1995 "In recognition of his contributions to the foundations of computational complexity theory and its application to cryptography and program checking". Some of his work includes the Blum Blum Shub pseudorandom number generator, the Blum-Goldwasser stream cypher, and more recently Captchas.
Blum attended MIT, where he received his Bachelor's degree in 1959, his Master's degree in 1961, and his PhD in 1964.
He worked as a professor of computer science at the University of California, Berkeley until 2000.
He is currently the Bruce Nelson Professor of Computer Science at Carnegie Mellon University.

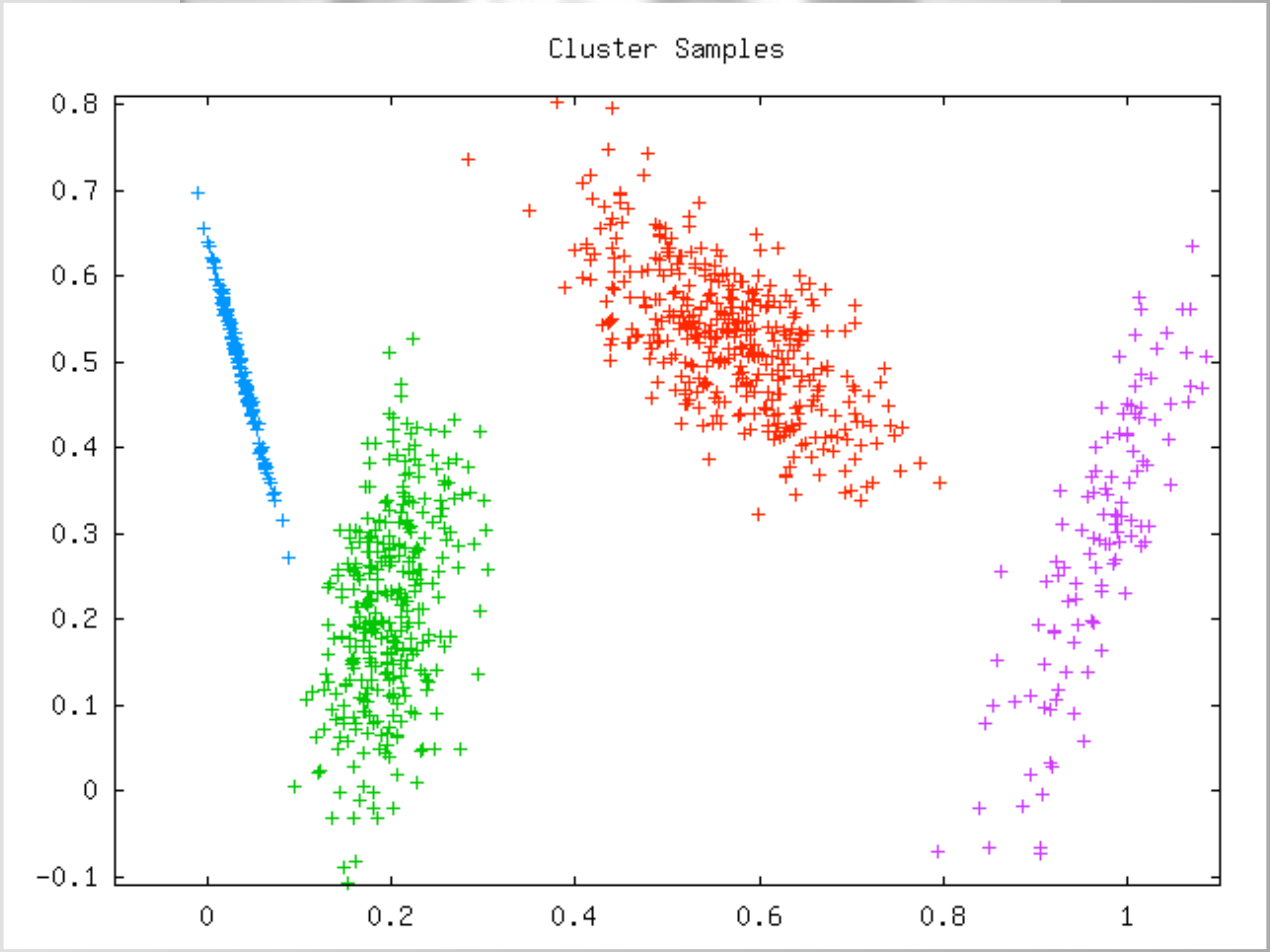
Recap: Unsupervised Learning


Data D $\{x_1,\ldots,x_N\}$
$x_i=(a_1,a_2,\ldots,a_M)^T$
x
Model $\{\theta_1, \theta_2, \ldots, \theta_L\}$
y

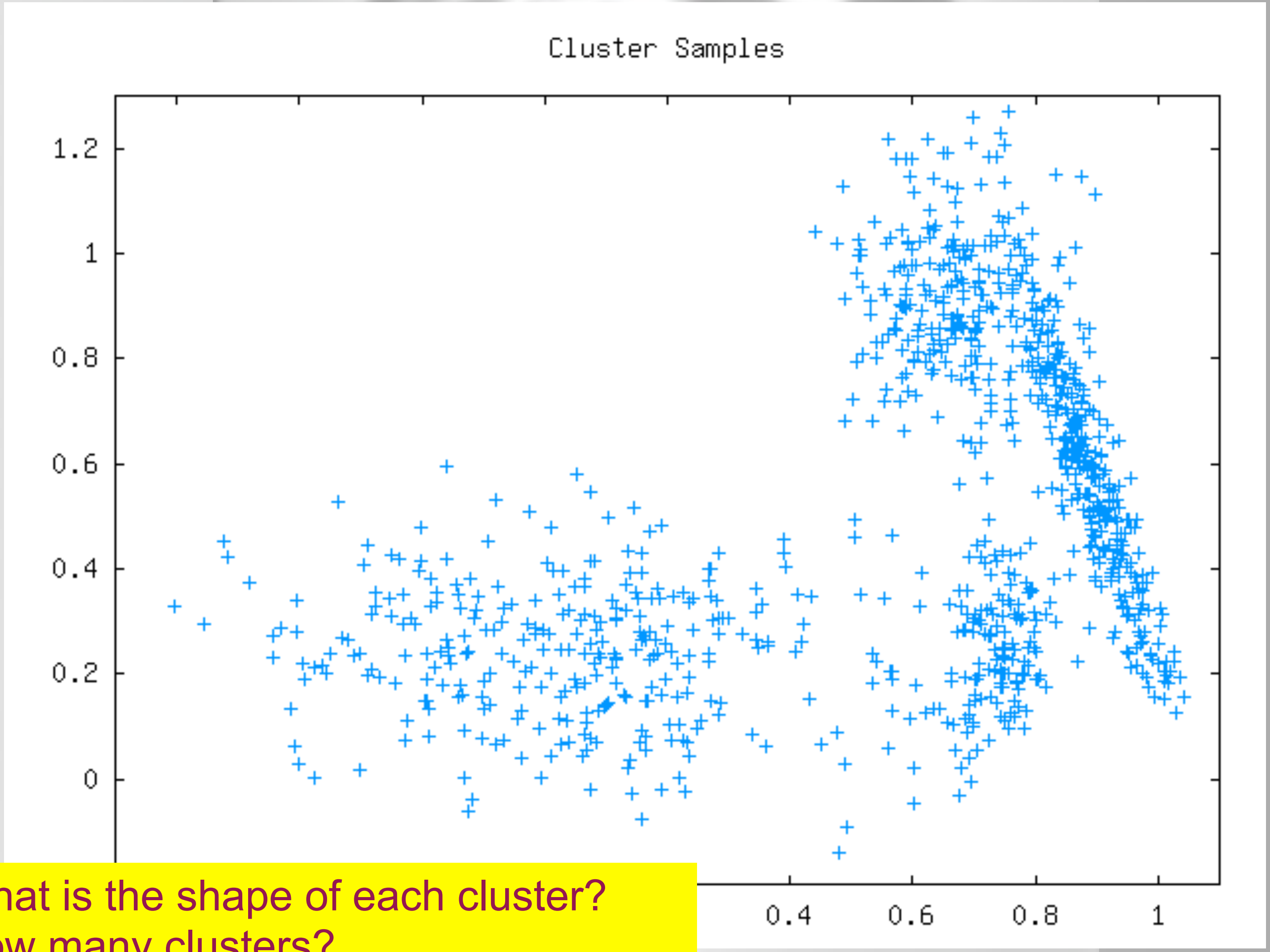
Clustering: y categorical

Dimensionality Reduction:
$x \in \Re^M \Rightarrow y \in \Re^K$, with K<M

Clustering

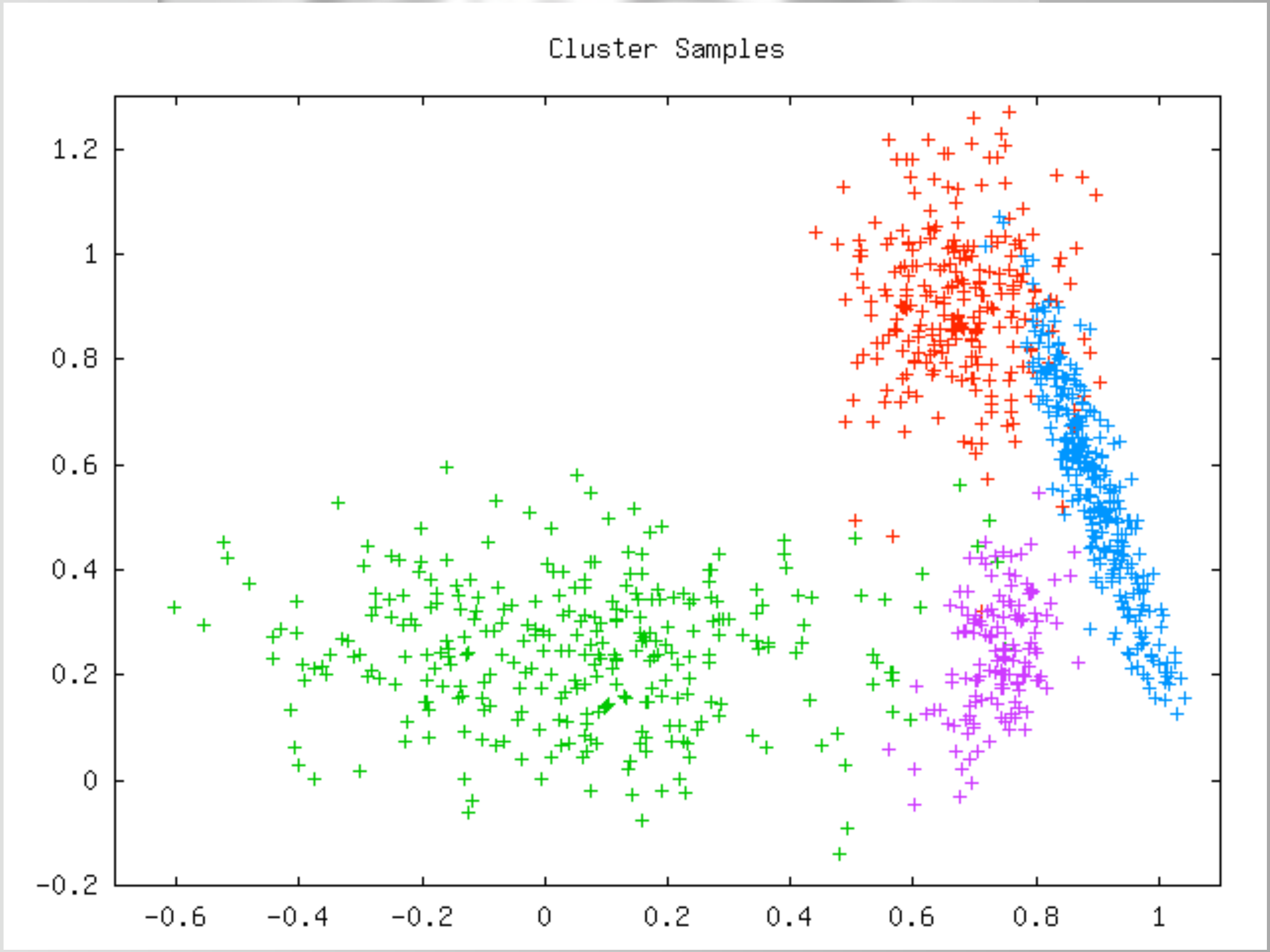


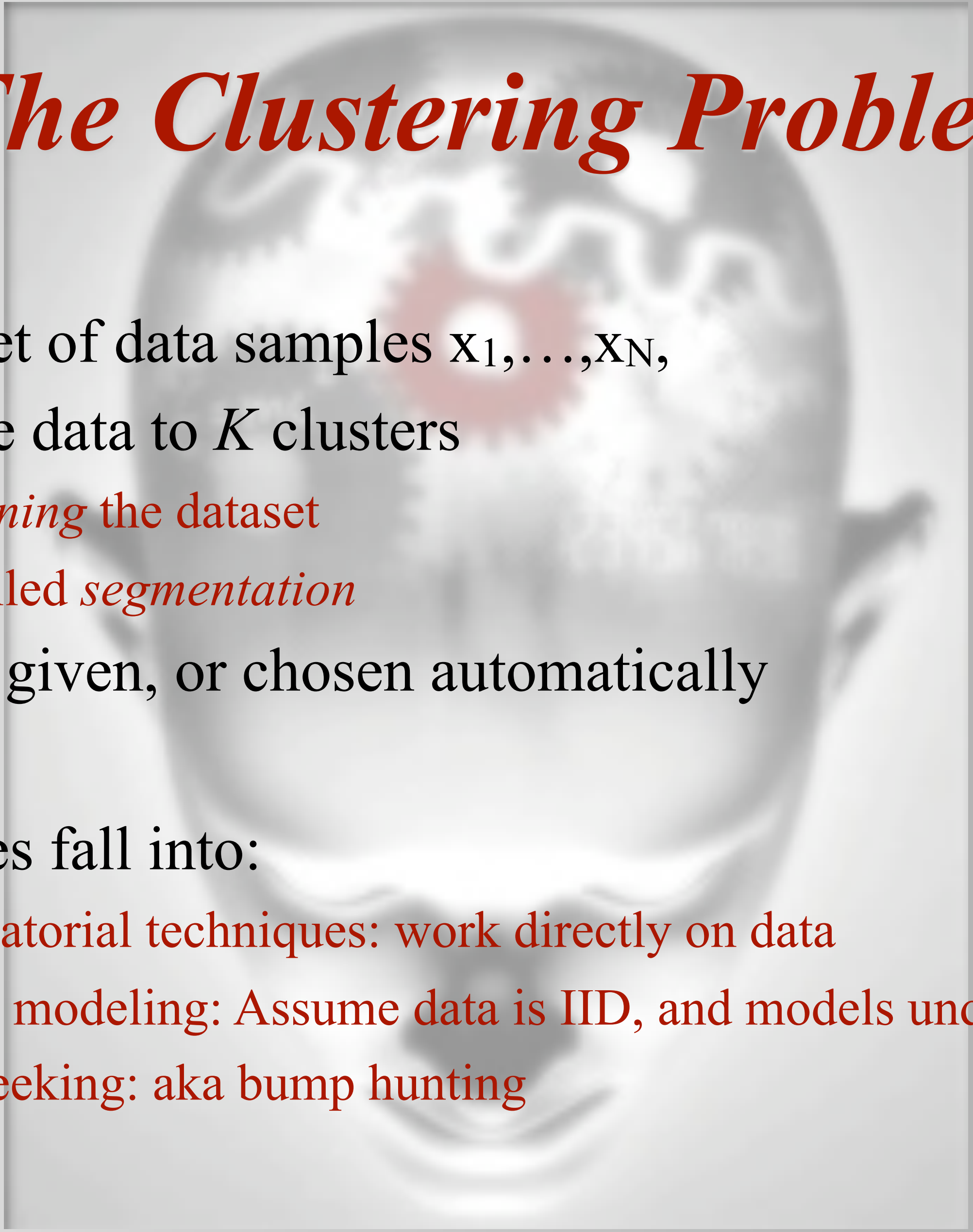
Where Are the Clusters?



- What is the shape of each cluster?
- How many clusters?

Clustering

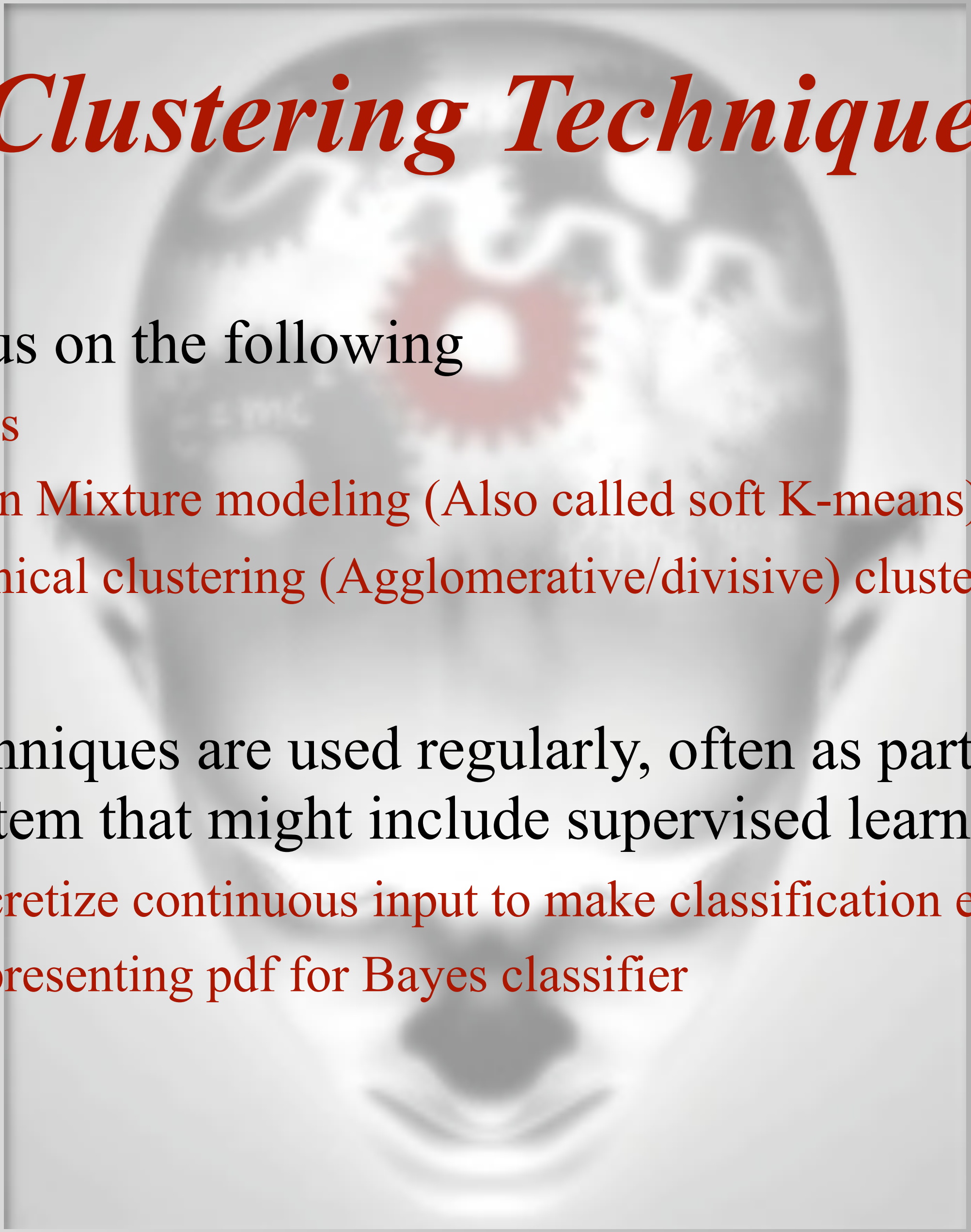




The Clustering Problem

- Given a set of data samples $x_1,\ldots,x_N$,
- Assign the data to *K* clusters
  - *Partitioning* the dataset
  - Also called *segmentation*
- *K* may be given, or chosen automatically

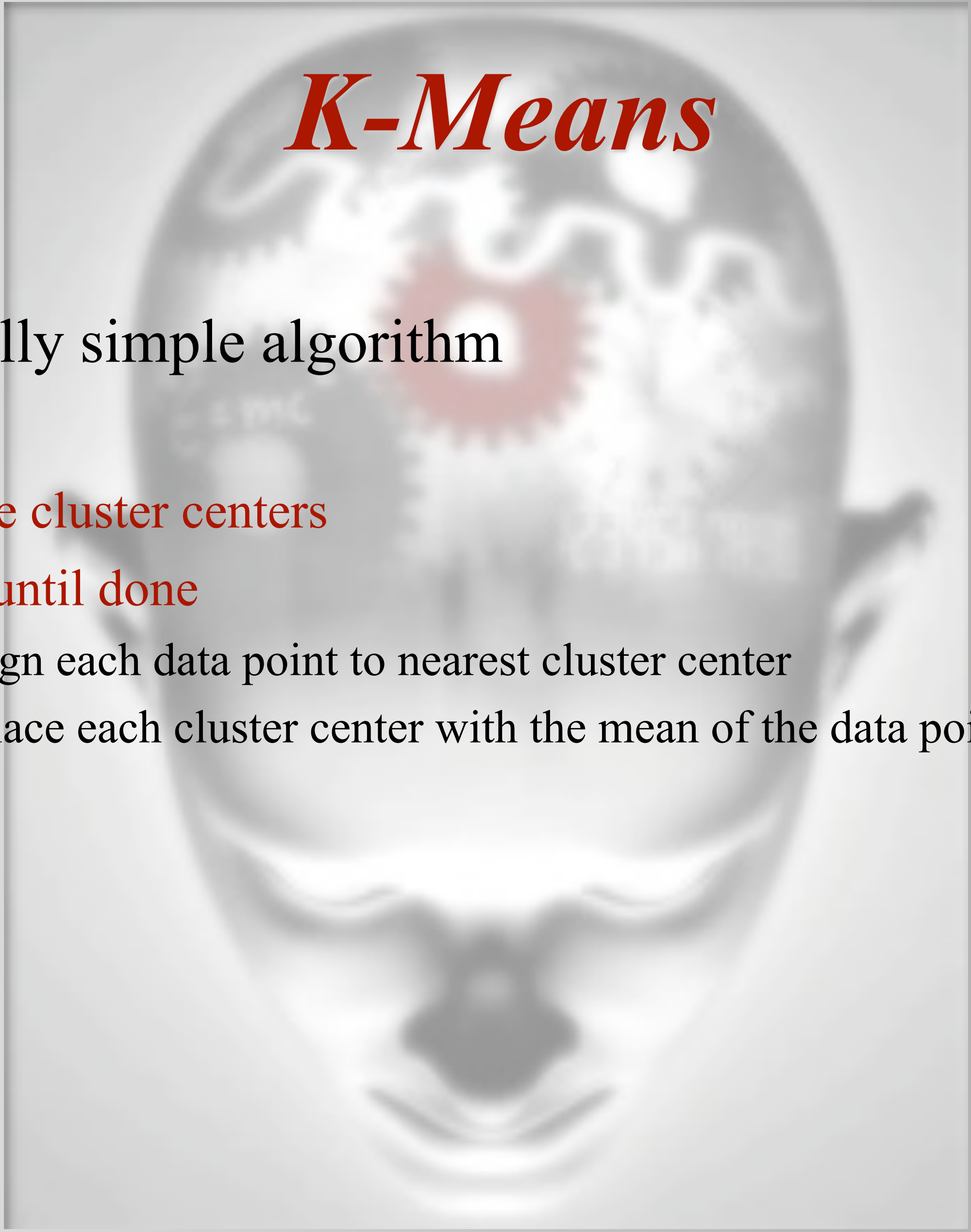
- Techniques fall into:
  - Combinatorial techniques: work directly on data
  - Mixture modeling: Assume data is IID, and models underlying pdf
  - Mode seeking: aka bump hunting



Clustering Techniques

- We'll focus on the following
  - K-means
  - Gaussian Mixture modeling (Also called soft K-means)
  - Hierarchical clustering (Agglomerative/divisive) clustering

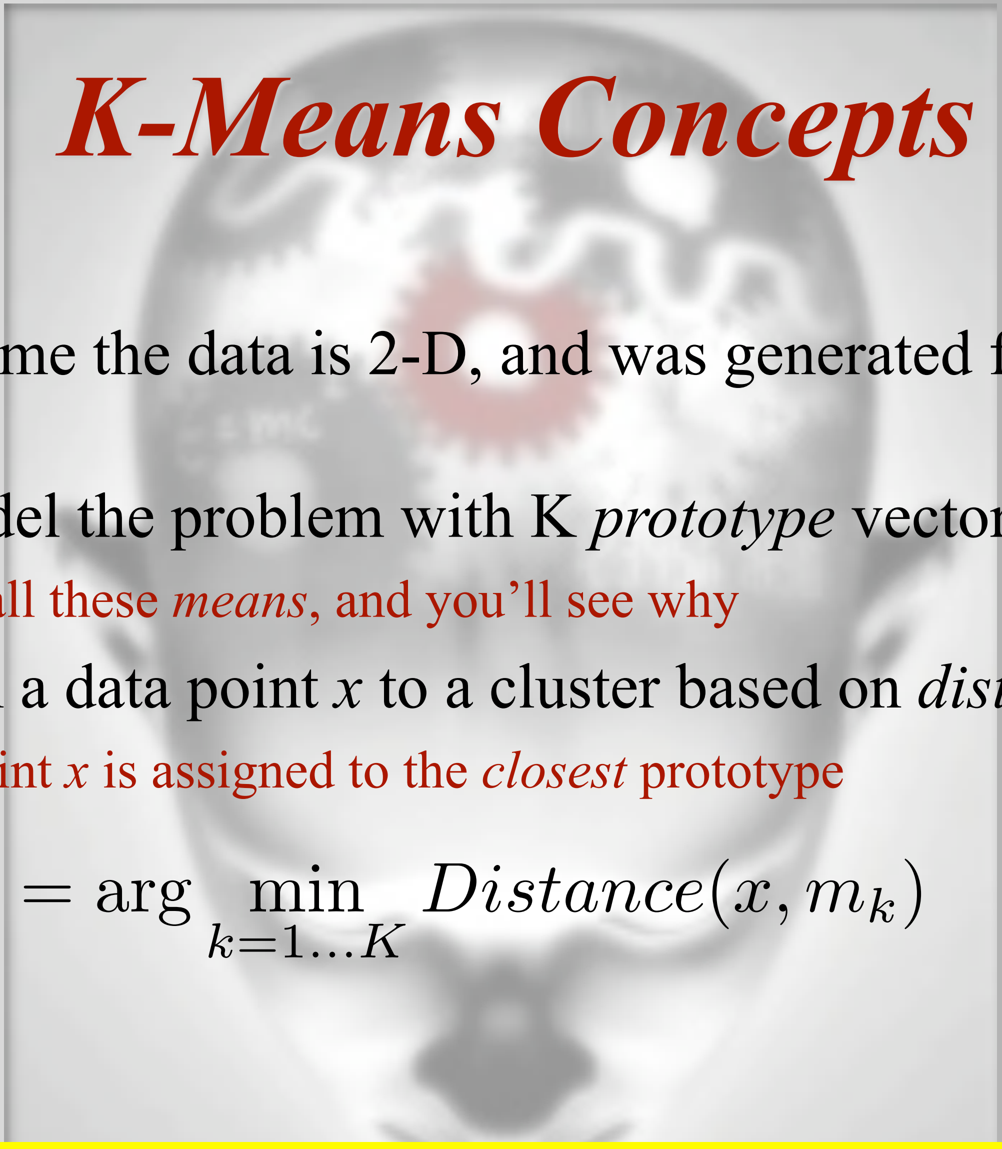
- These techniques are used regularly, often as part of a much larger system that might include supervised learning
  - e.g. discretize continuous input to make classification easier
  - e.g. Representing pdf for Bayes classifier



K-Means

- Wonderfully simple algorithm
- K-means:
  - Initialize cluster centers
  - Repeat until done
    - Assign each data point to nearest cluster center
    - Replace each cluster center with the mean of the data points associated to it

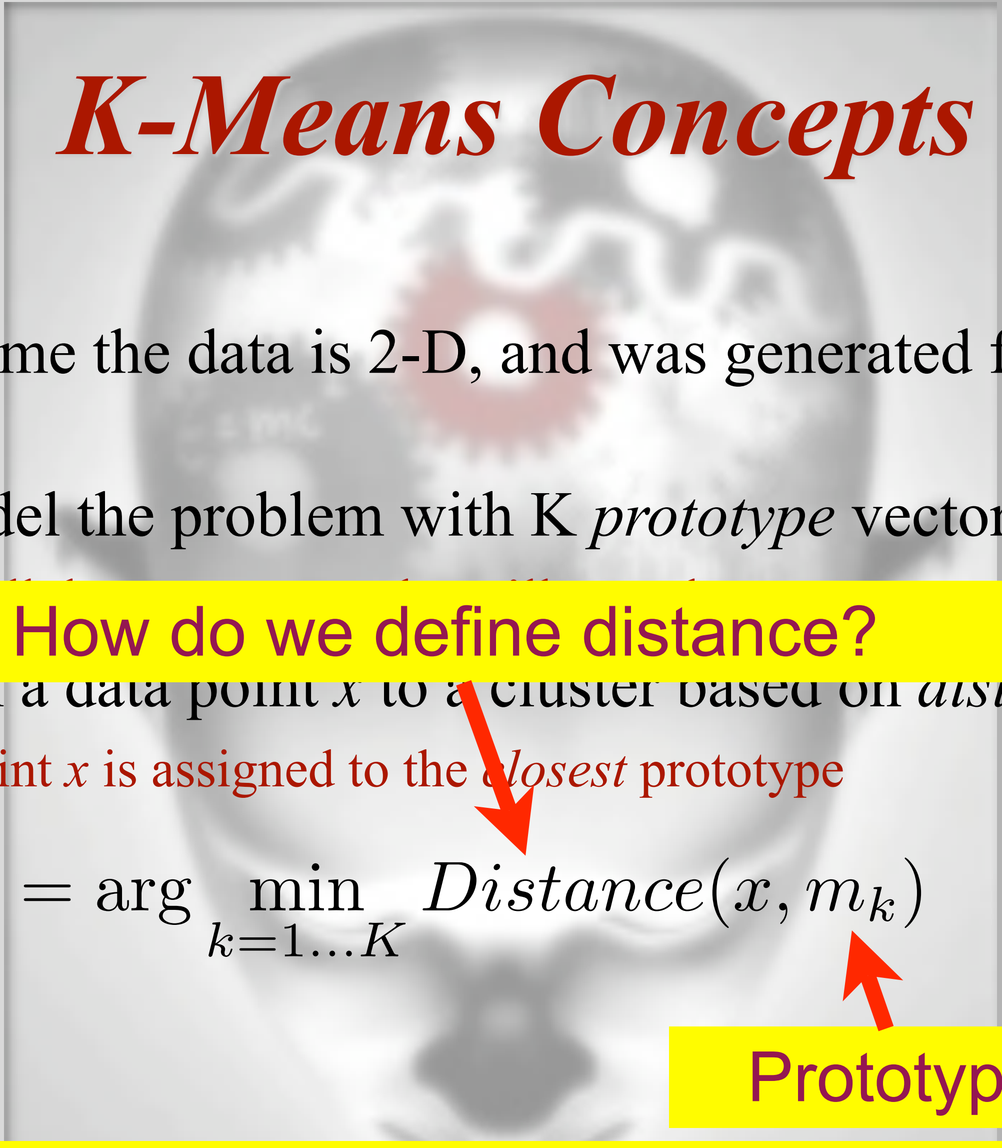
K-Means Concepts



- Let's assume the data is 2-D, and was generated from K clusters
- We'll model the problem with K *prototype* vectors
  - We'll call these *means*, and you'll see why
- We assign a data point x to a cluster based on *distance*
  - Data point x is assigned to the *closest* prototype

$$y = \arg \min_{k=1 \dots K} Distance(x, m_k)$$

Note, this is similar to nearest neighbor classification and regression methods, which we will come back to



K-Means Concepts

- Let's assume the data is 2-D, and was generated from K clusters
- We'll model the problem with K *prototype* vectors
  - We

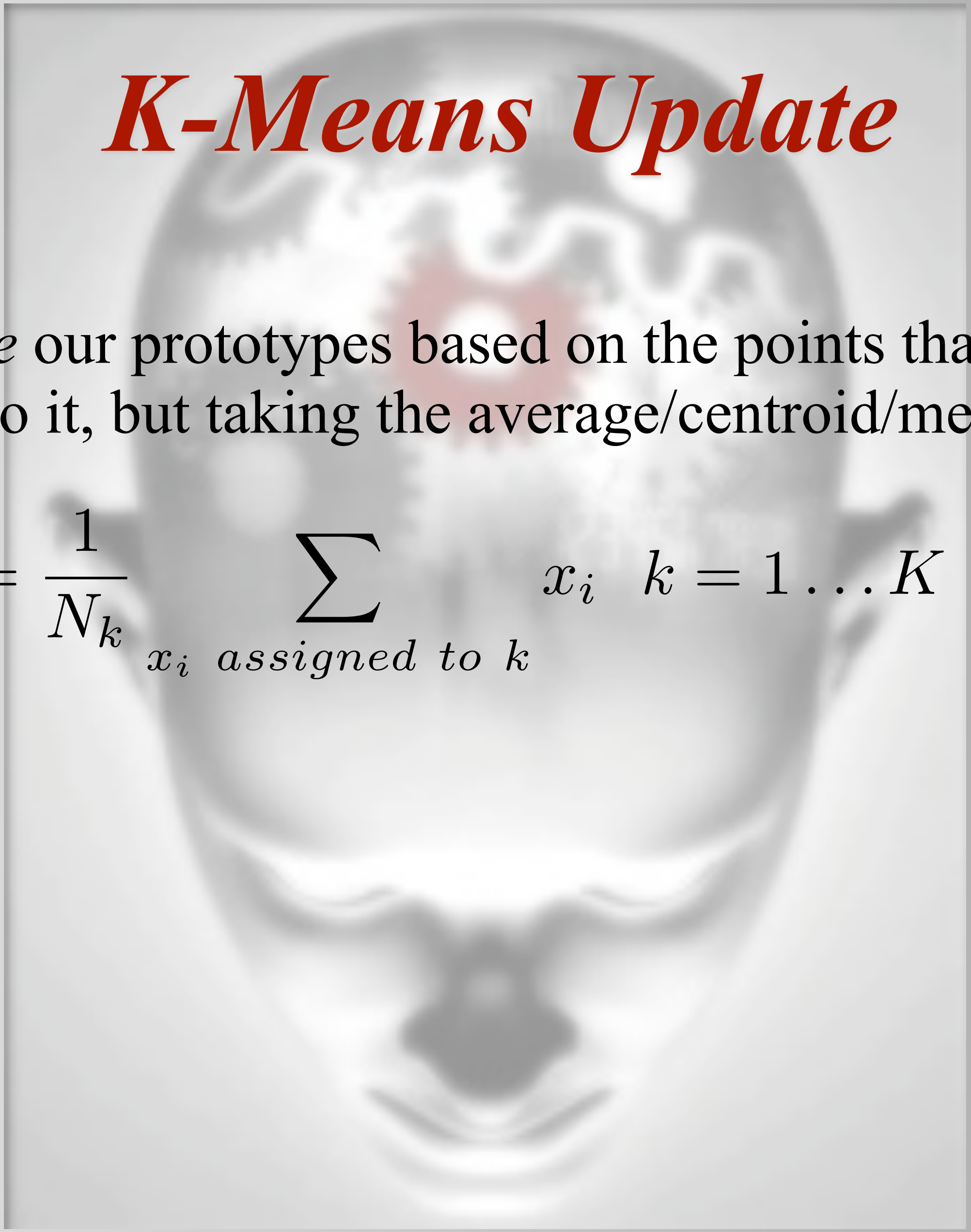
How do we define distance?

- We assign a data point *x* to a cluster based on *distance*
  - Data point *x* is assigned to the *closest* prototype

$$y = \arg \min_{k=1 \ldots K} Distance(x, m_k)$$

Prototypes

Note, this is similar to nearest neighbor classification and regression methods, which we will come back to



K-Means Update

- We *update* our prototypes based on the points that were assigned to it, but taking the average/centroid/mean

$$m'_k = \frac{1}{N_k} \sum_{x_i \ assigned \ to \ k} x_i \quad k = 1 \ldots K$$

K-Means Update

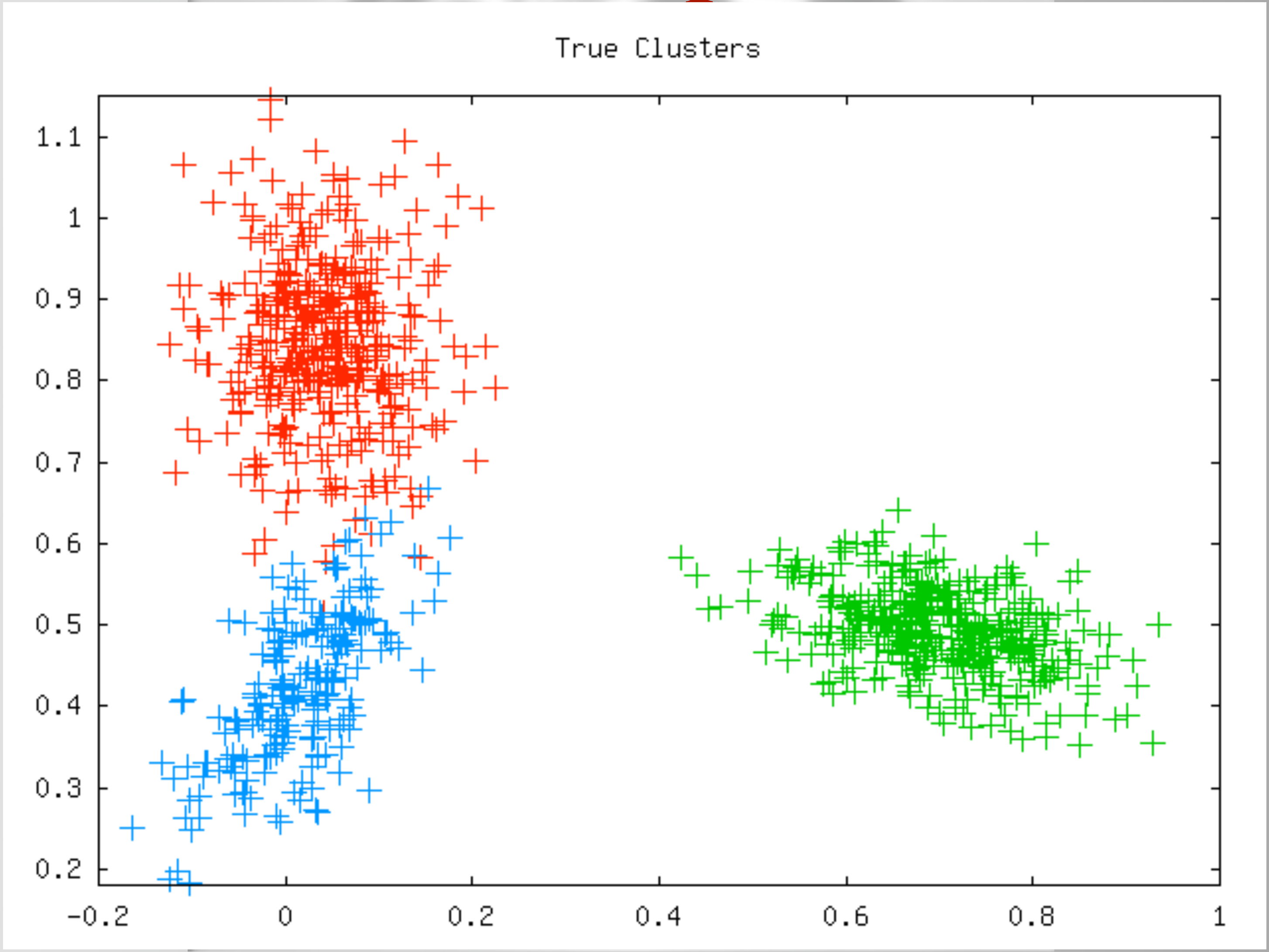
- We *update* our prototypes based on the points that were assigned to it, but taking the average/centroid/mean

$$m'_k = \frac{1}{N_k} \sum_{x_i \text{ assigned to } k} x_i \quad k = 1 \ldots K$$

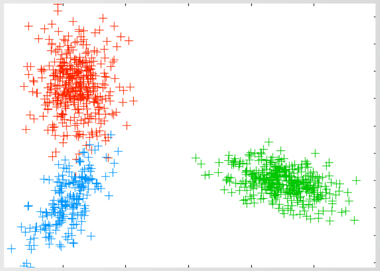
New prototype

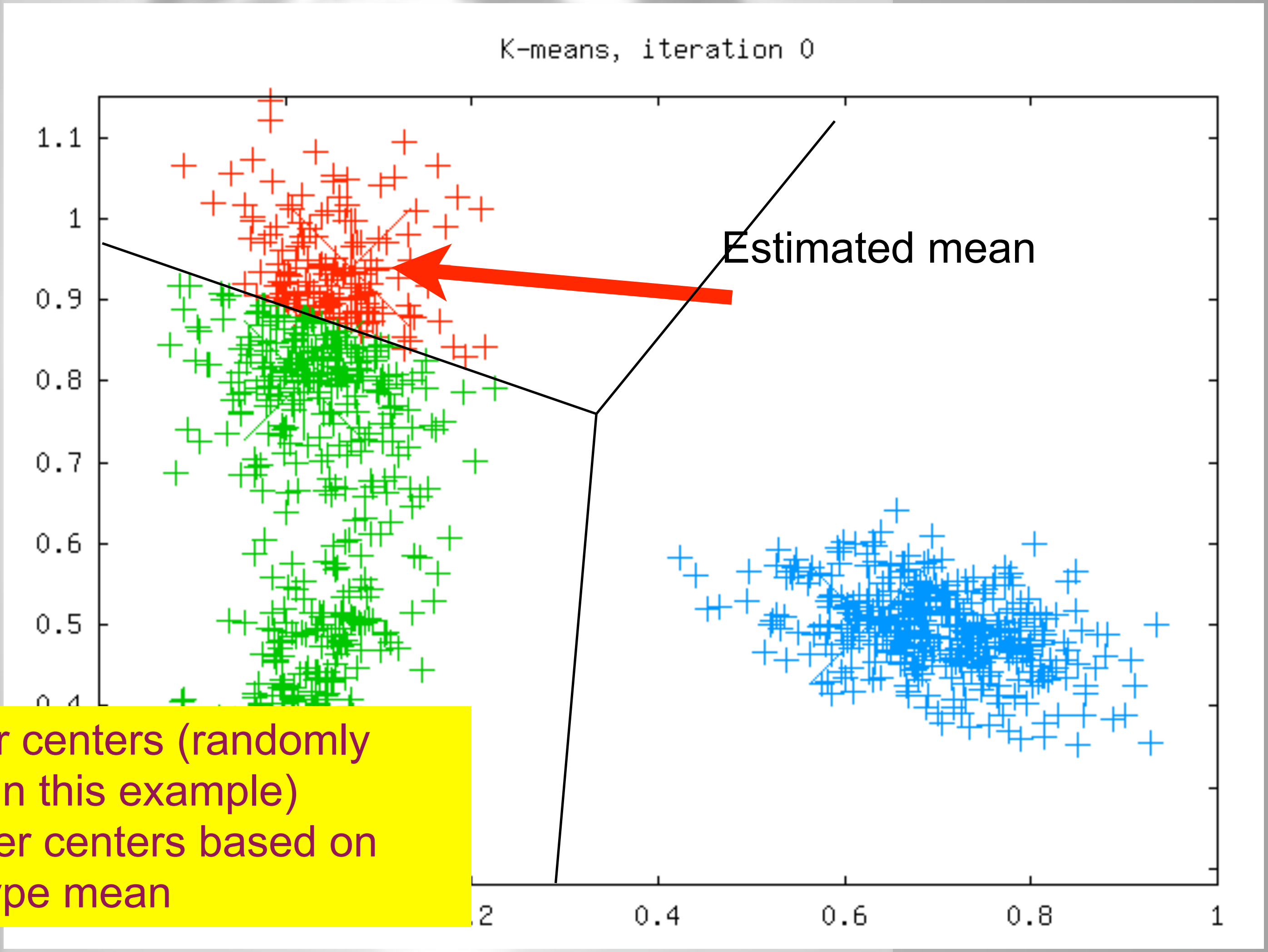
Mean of points assigned to *k*

Example



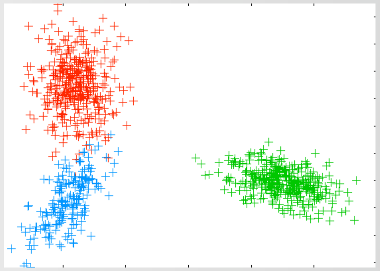
Example: Initialization

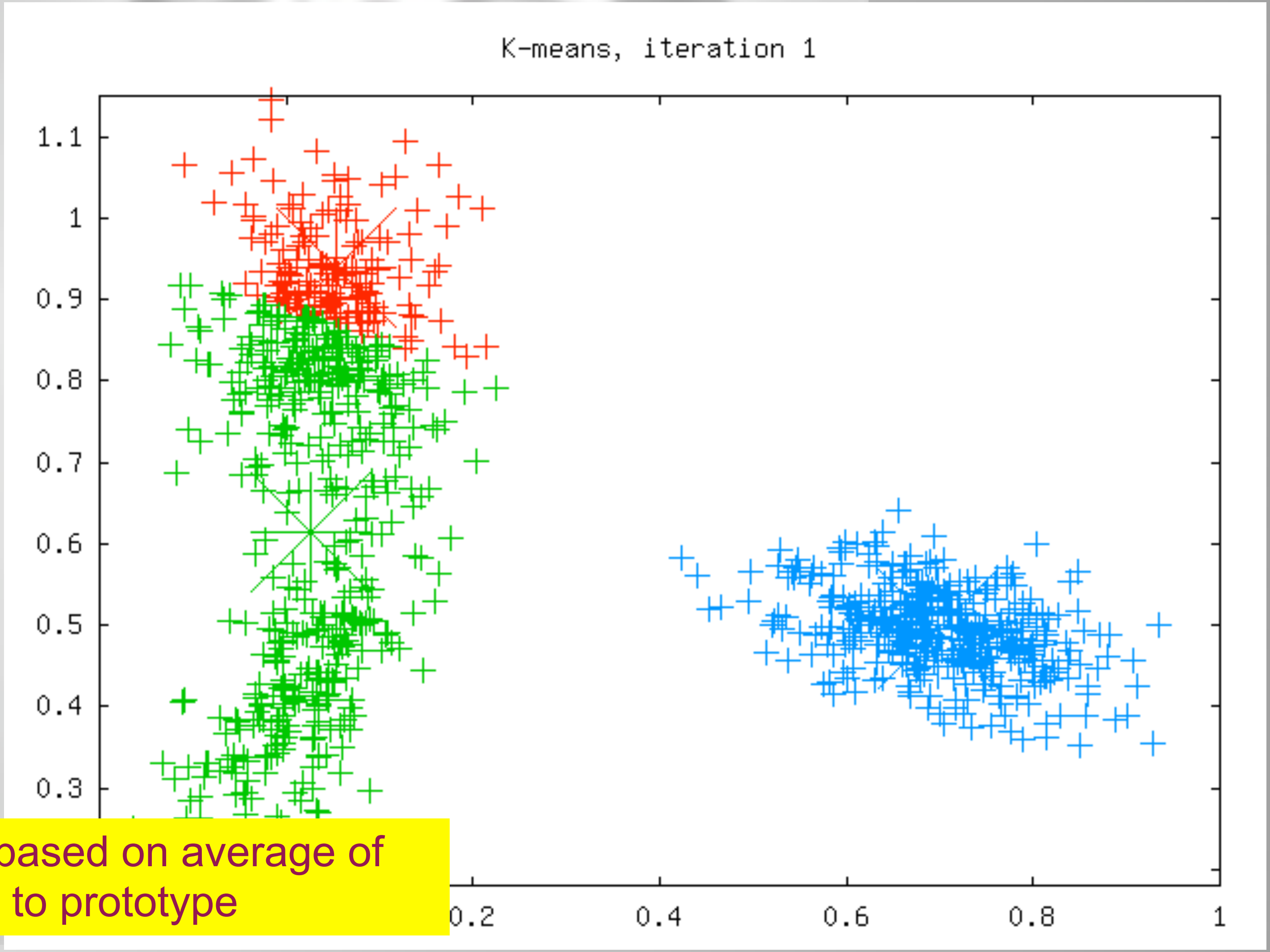




- Initialize cluster centers (randomly selecting data in this example)
- Assign to cluster centers based on nearest prototype mean

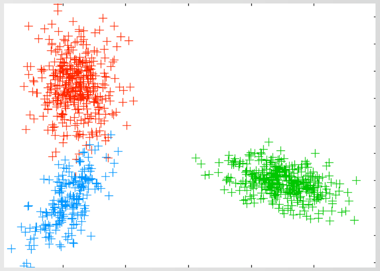
Example: 1 Iteration

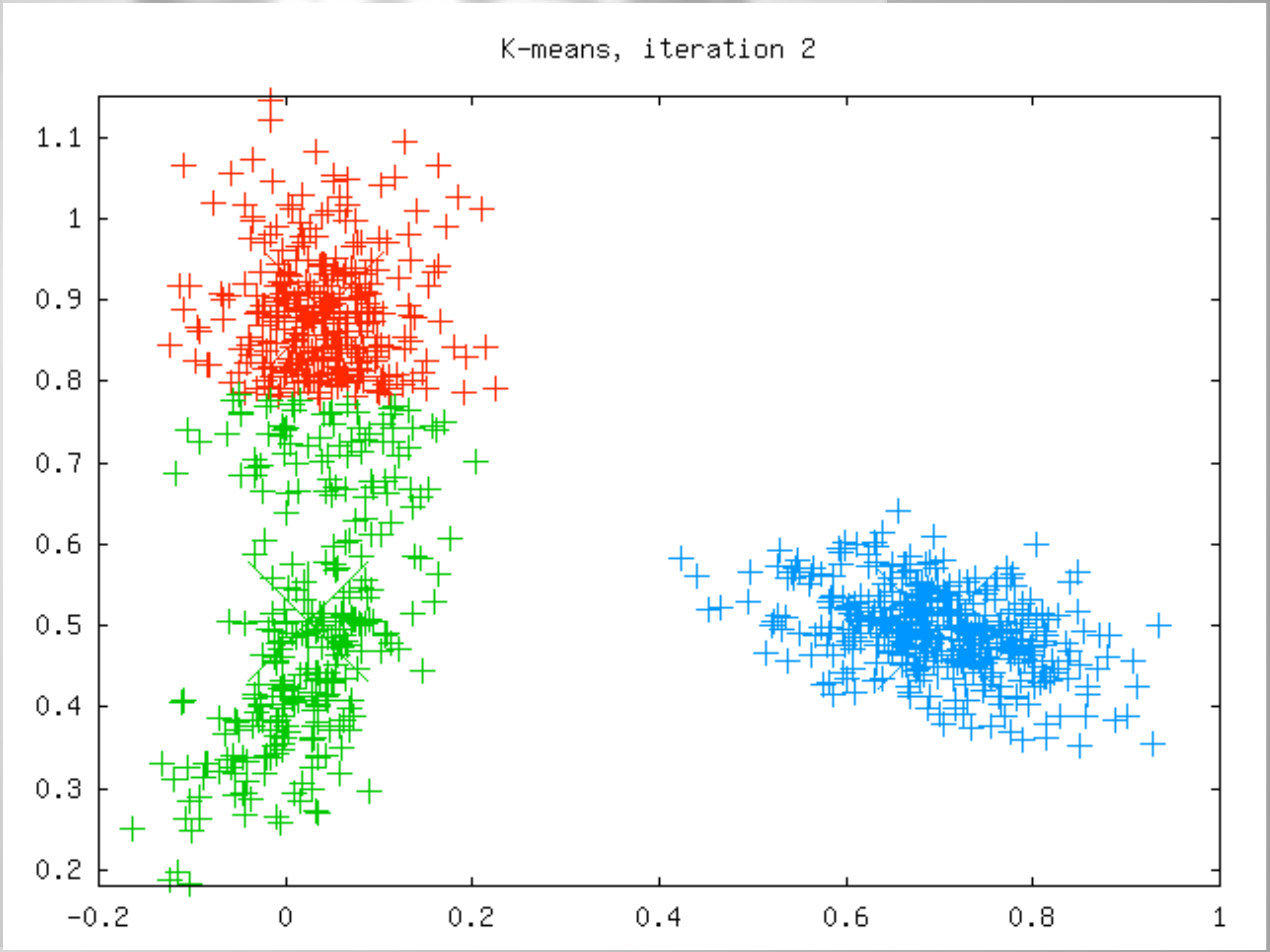




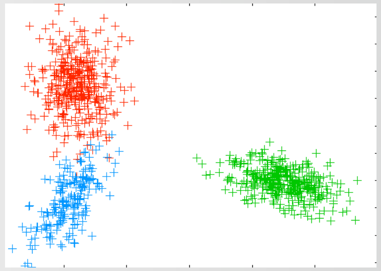
- Update means based on average of points assigned to prototype

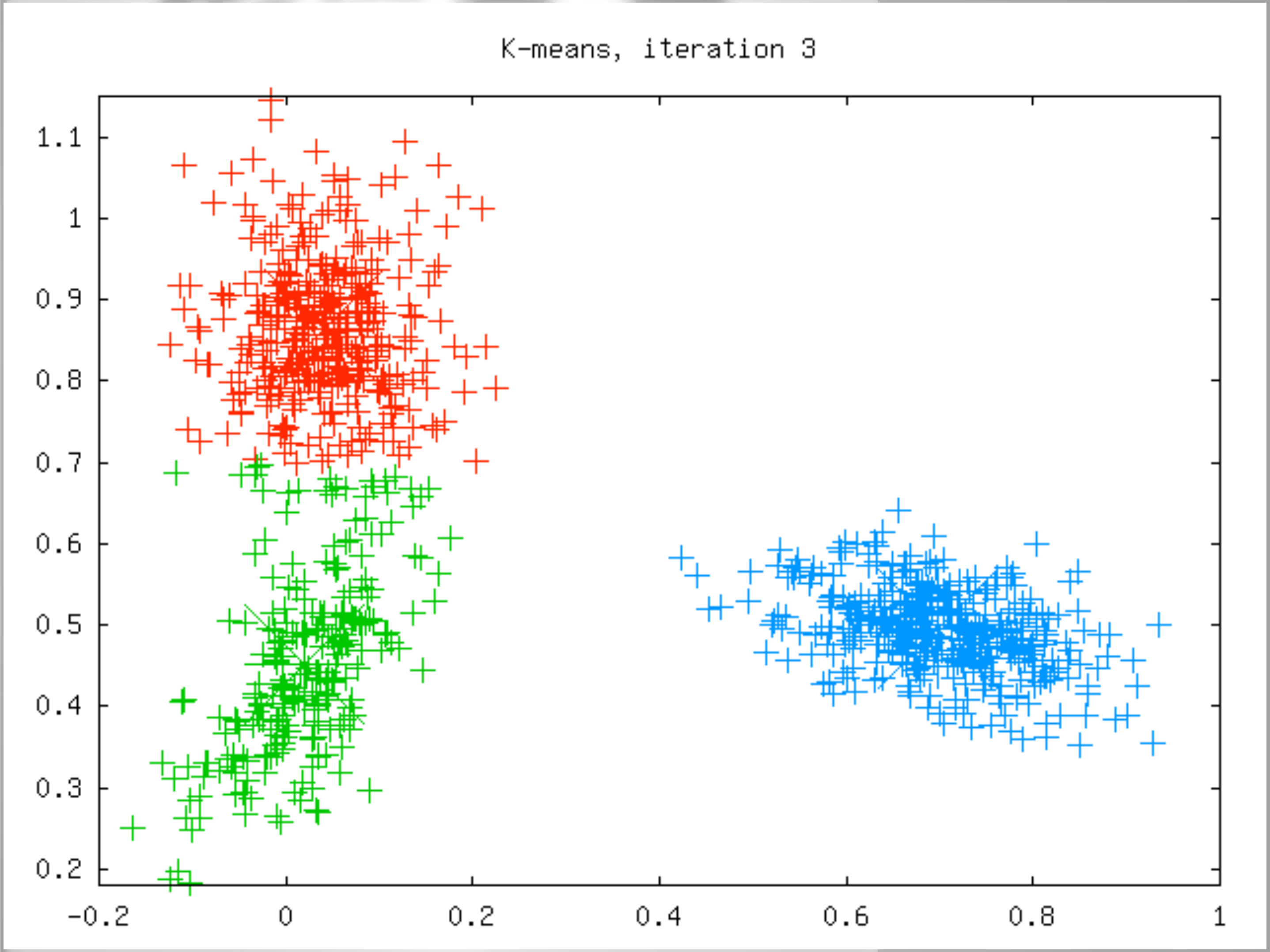
Iteration 2



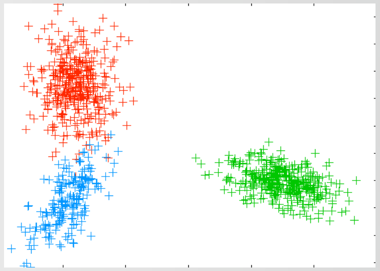


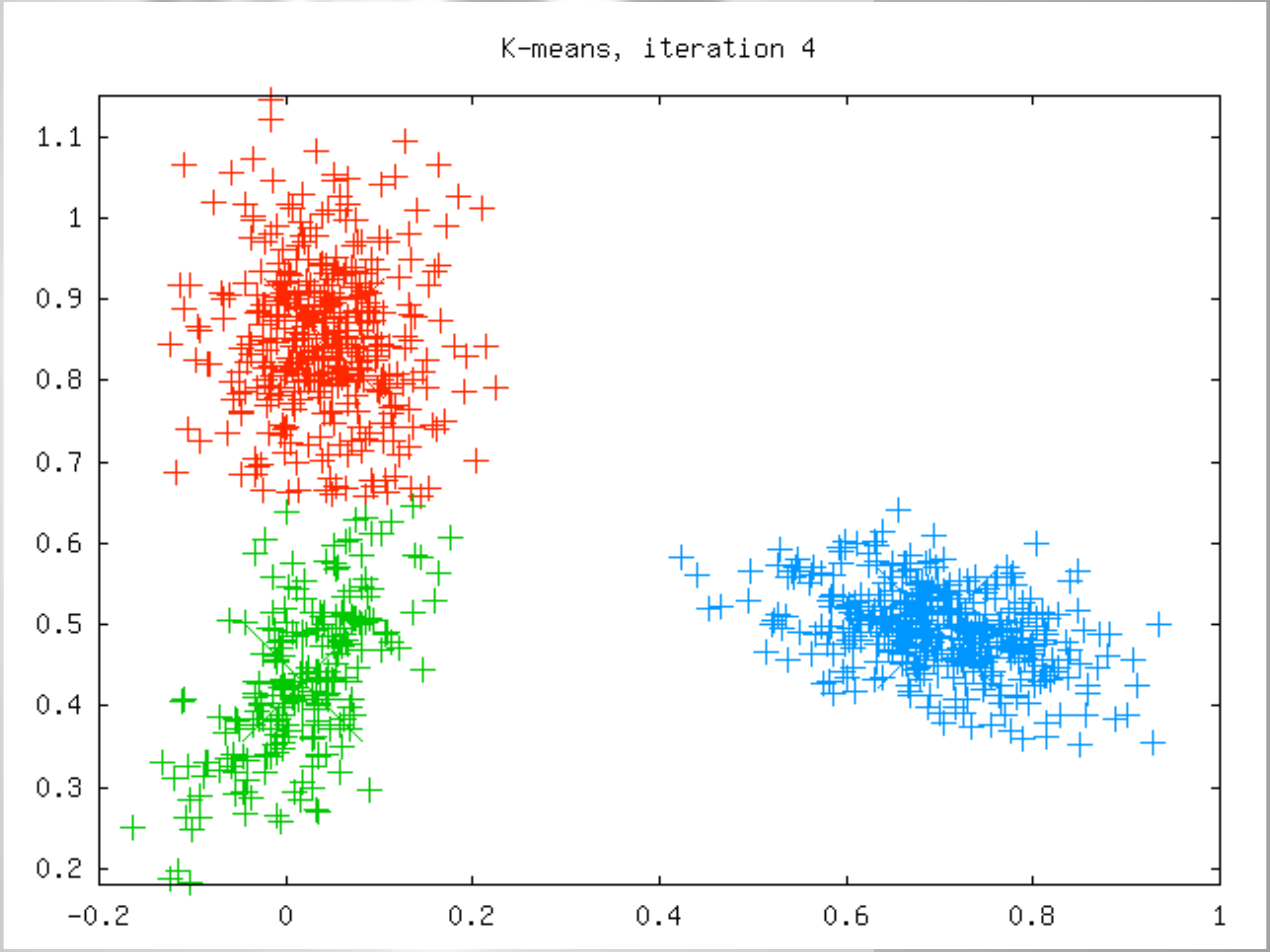
Iteration 3



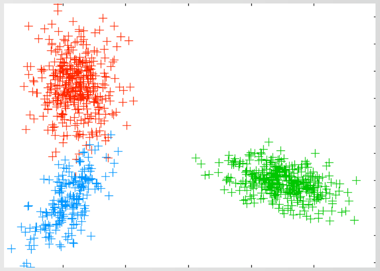


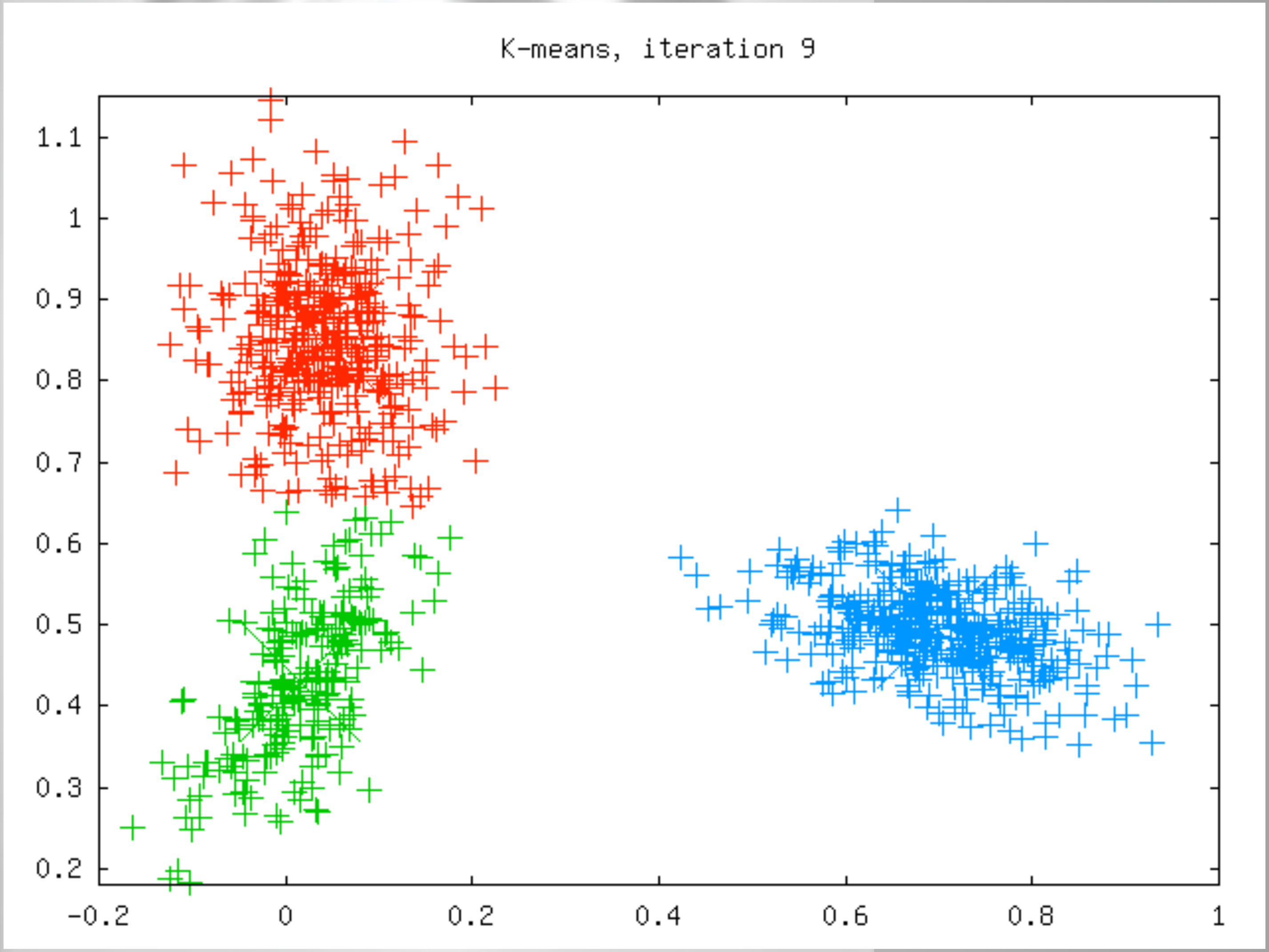
Iteration 4

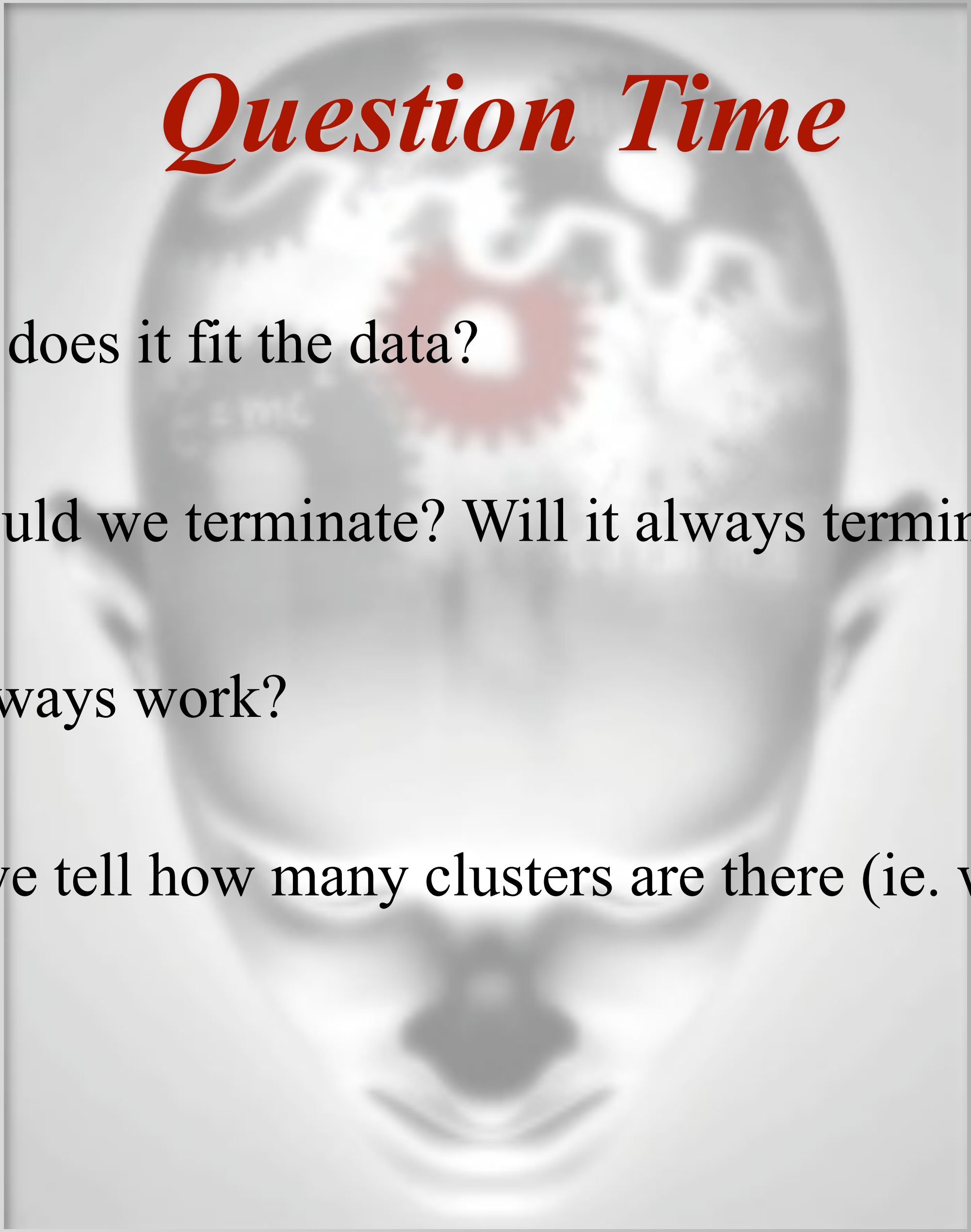




Iteration 9

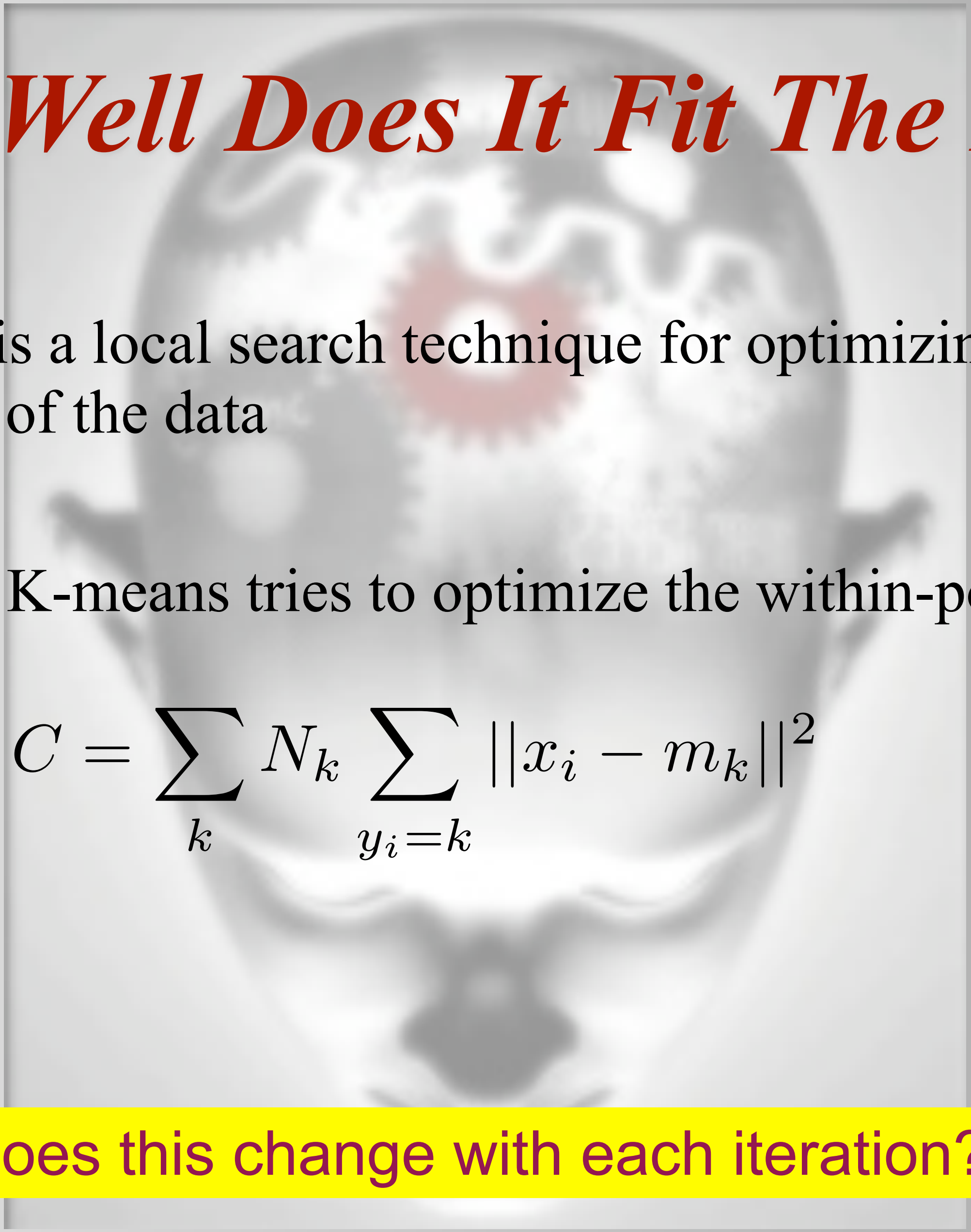






Question Time

- How *well* does it fit the data?
- When should we terminate? Will it always terminate?
- Does it always work?
- How do we tell how many clusters are there (ie. what is K)?



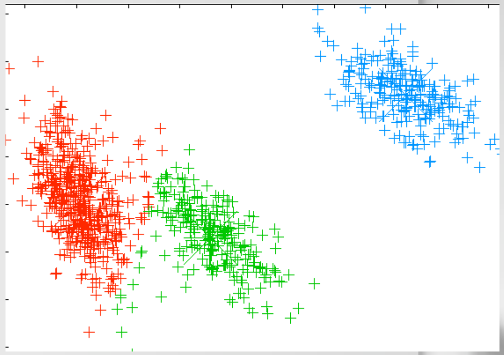
How Well Does It Fit The Data?

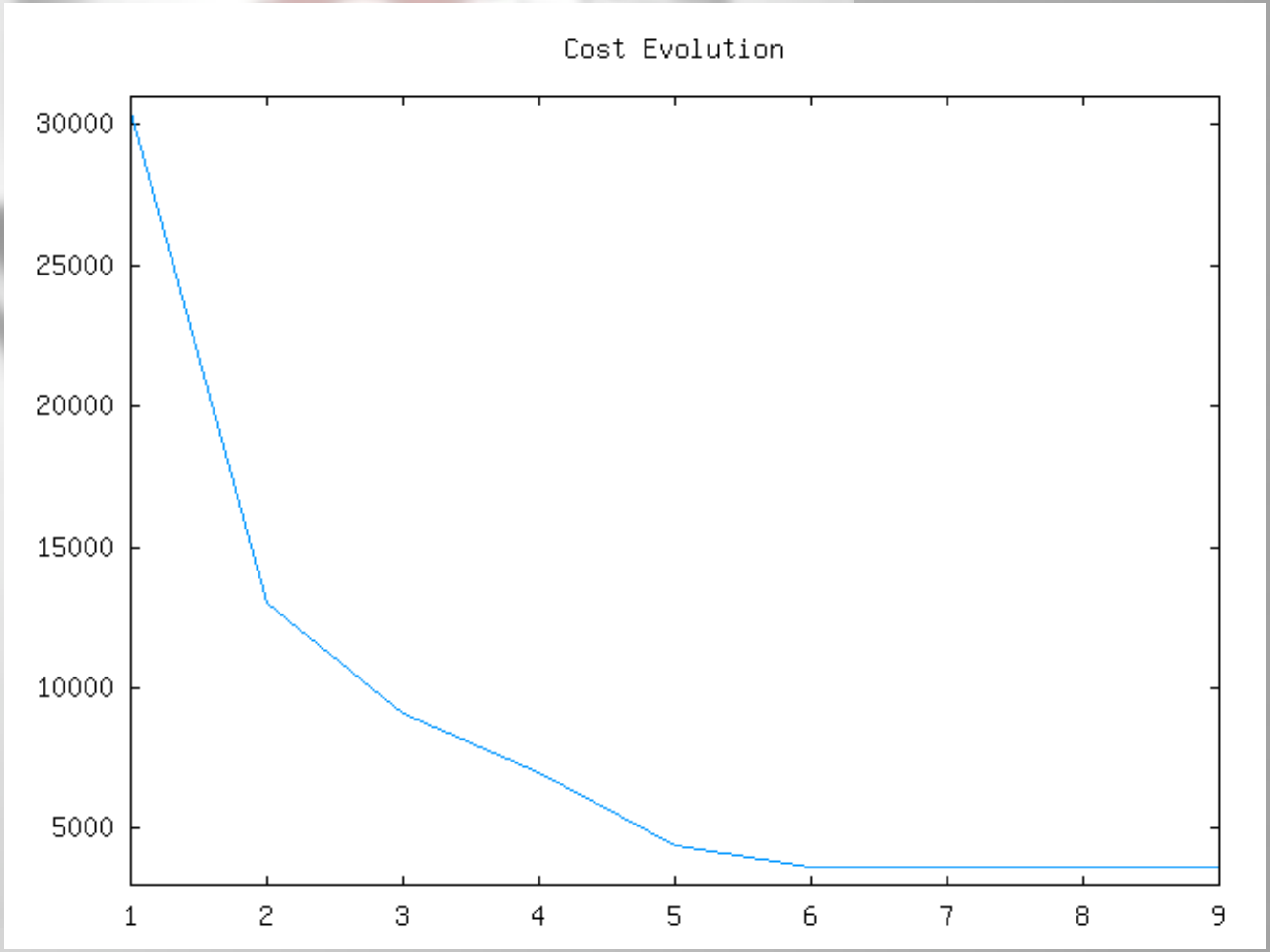
- K-means is a local search technique for optimizing the distortion of the data
- Formally, K-means tries to optimize the within-point scatter

$$C = \sum_k N_k \sum_{y_i = k} ||x_i - m_k||^2$$

How does this change with each iteration?

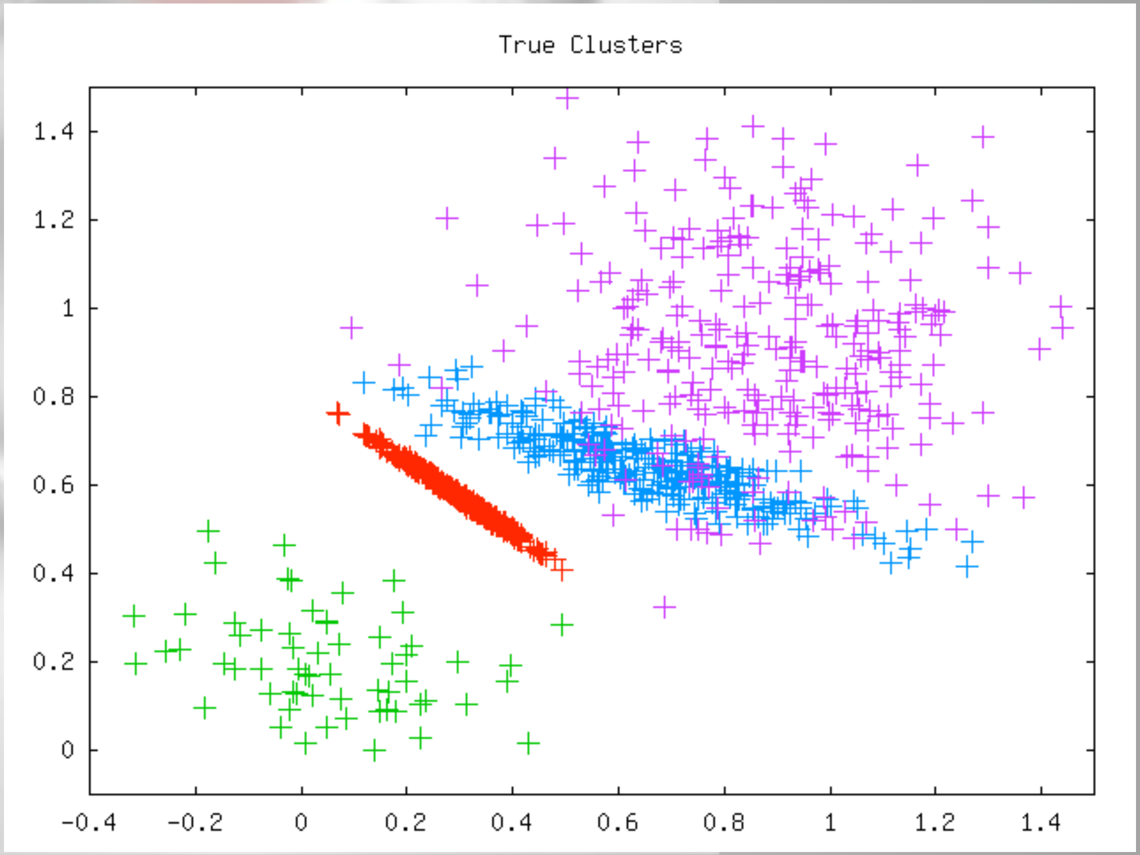
From Previous Example



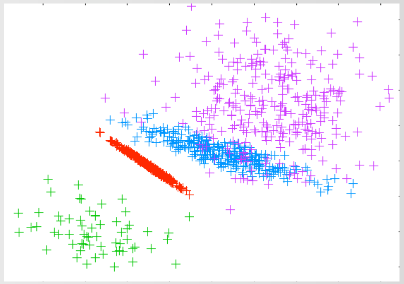


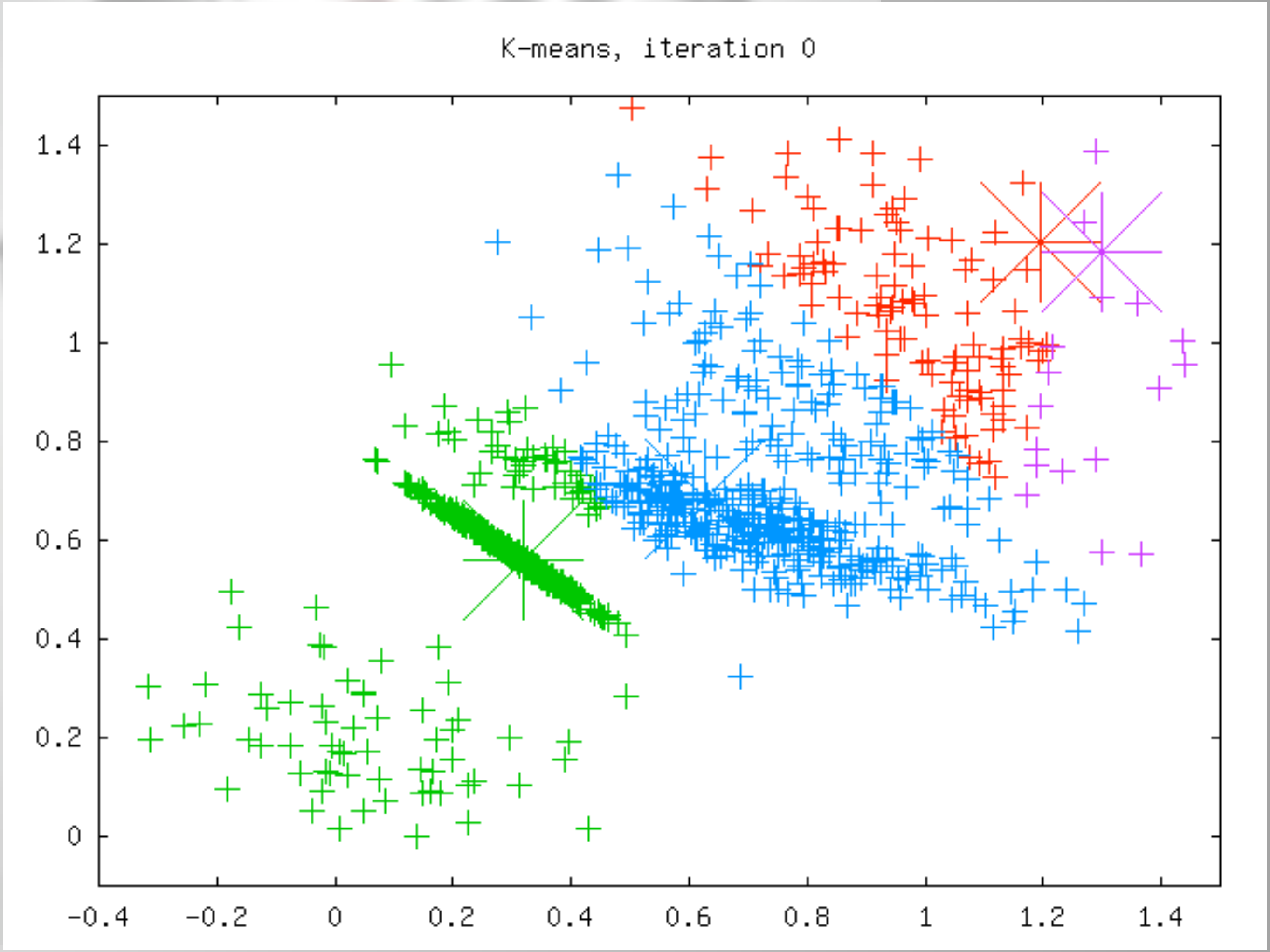
Does It Always Work?

- Unfortunately, no

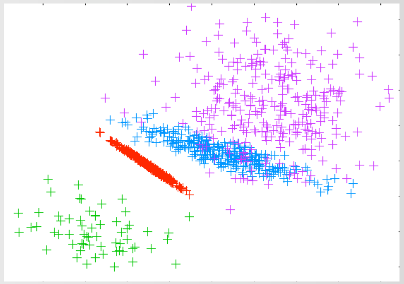


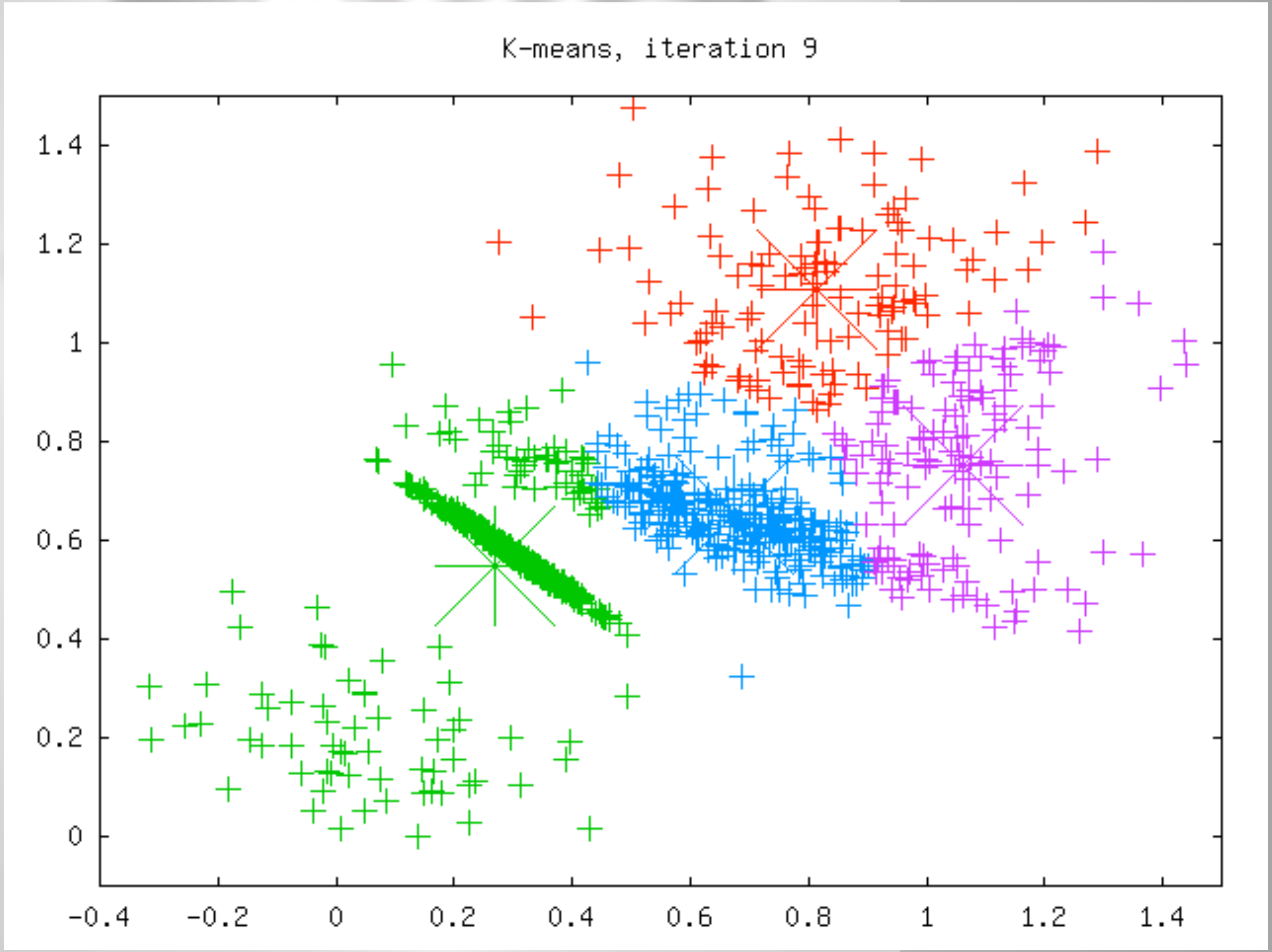
Does It Always Work?

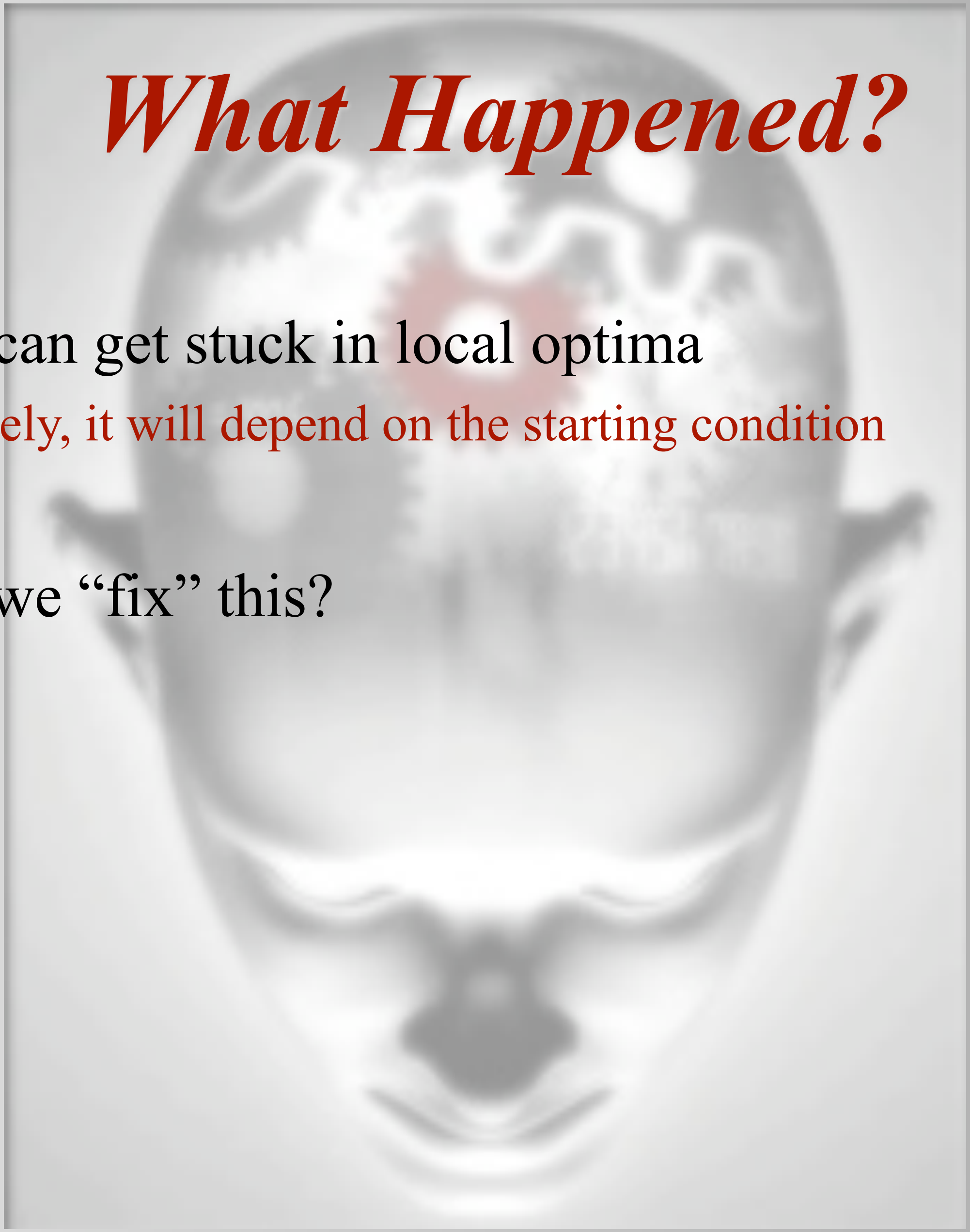




After 9 Iterations



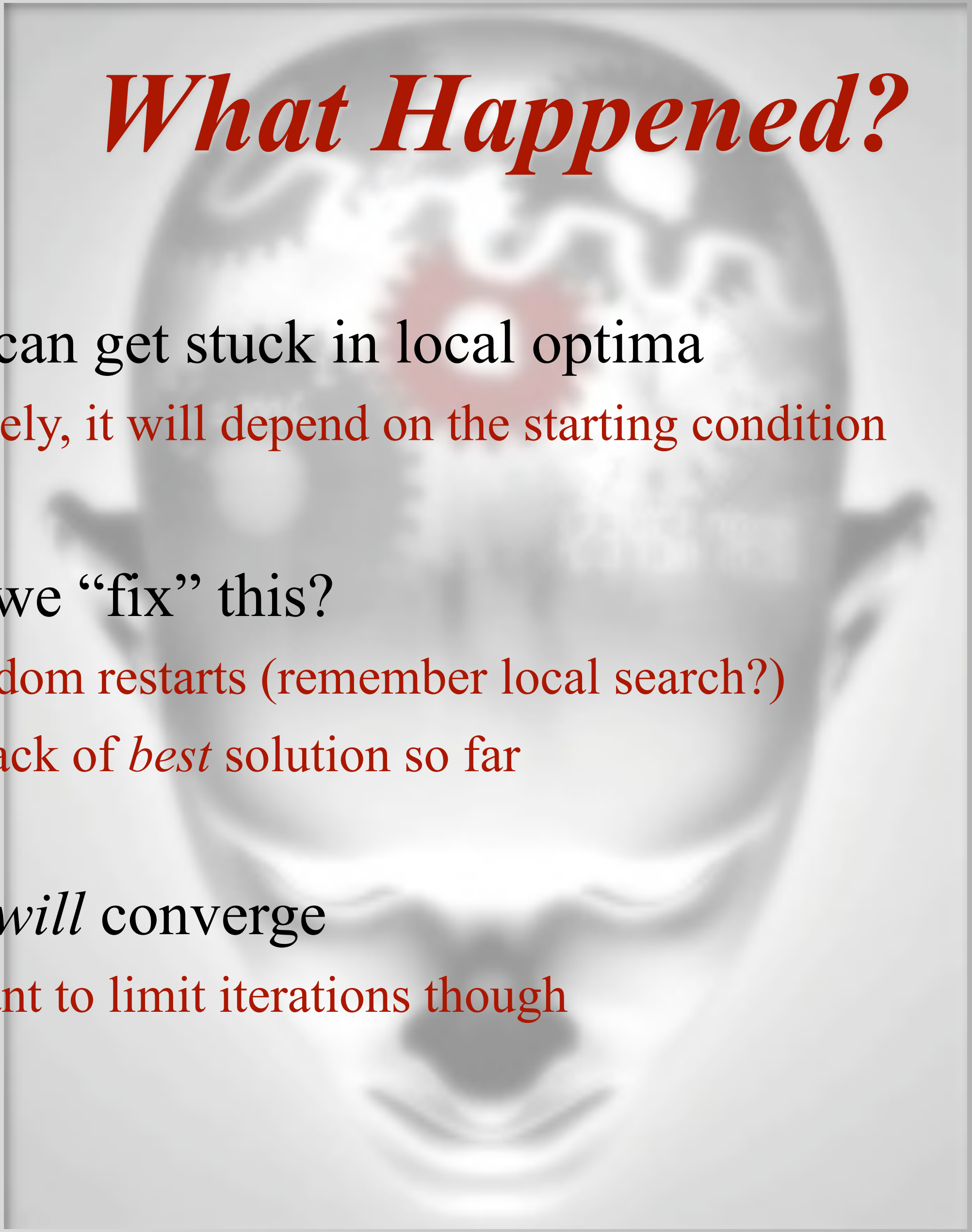




What Happened?

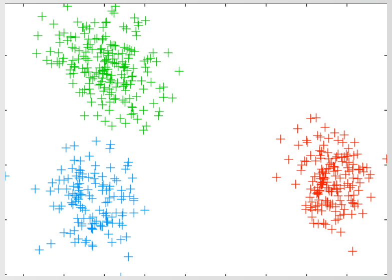
- K-means can get stuck in local optima
  - Effectively, it will depend on the starting condition
- How can we "fix" this?

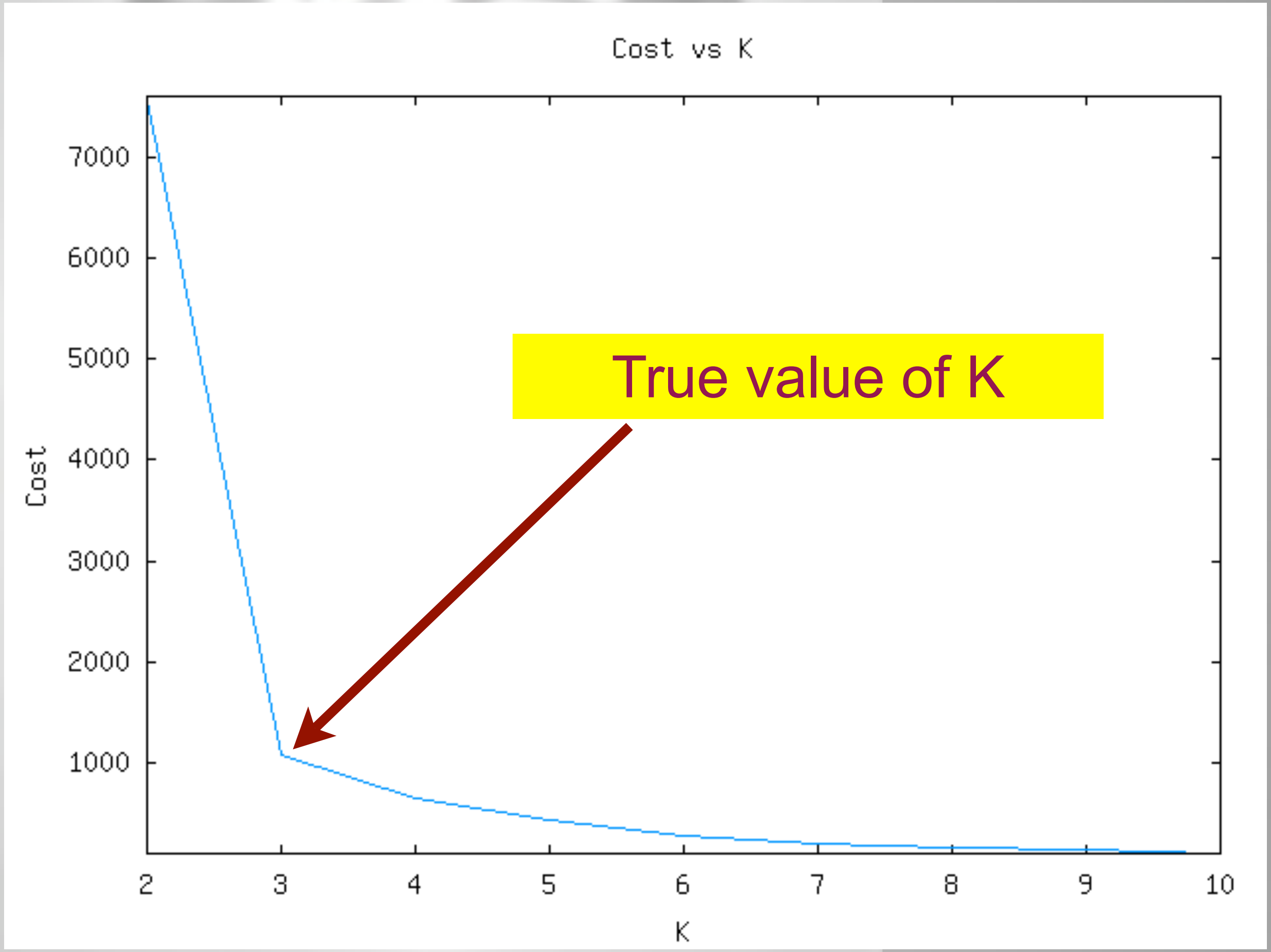
What Happened?



- K-means can get stuck in local optima
  - Effectively, it will depend on the starting condition
- How can we “fix” this?
  - Use random restarts (remember local search?)
  - Keep track of *best* solution so far
- K-means *will* converge
  - May want to limit iterations though

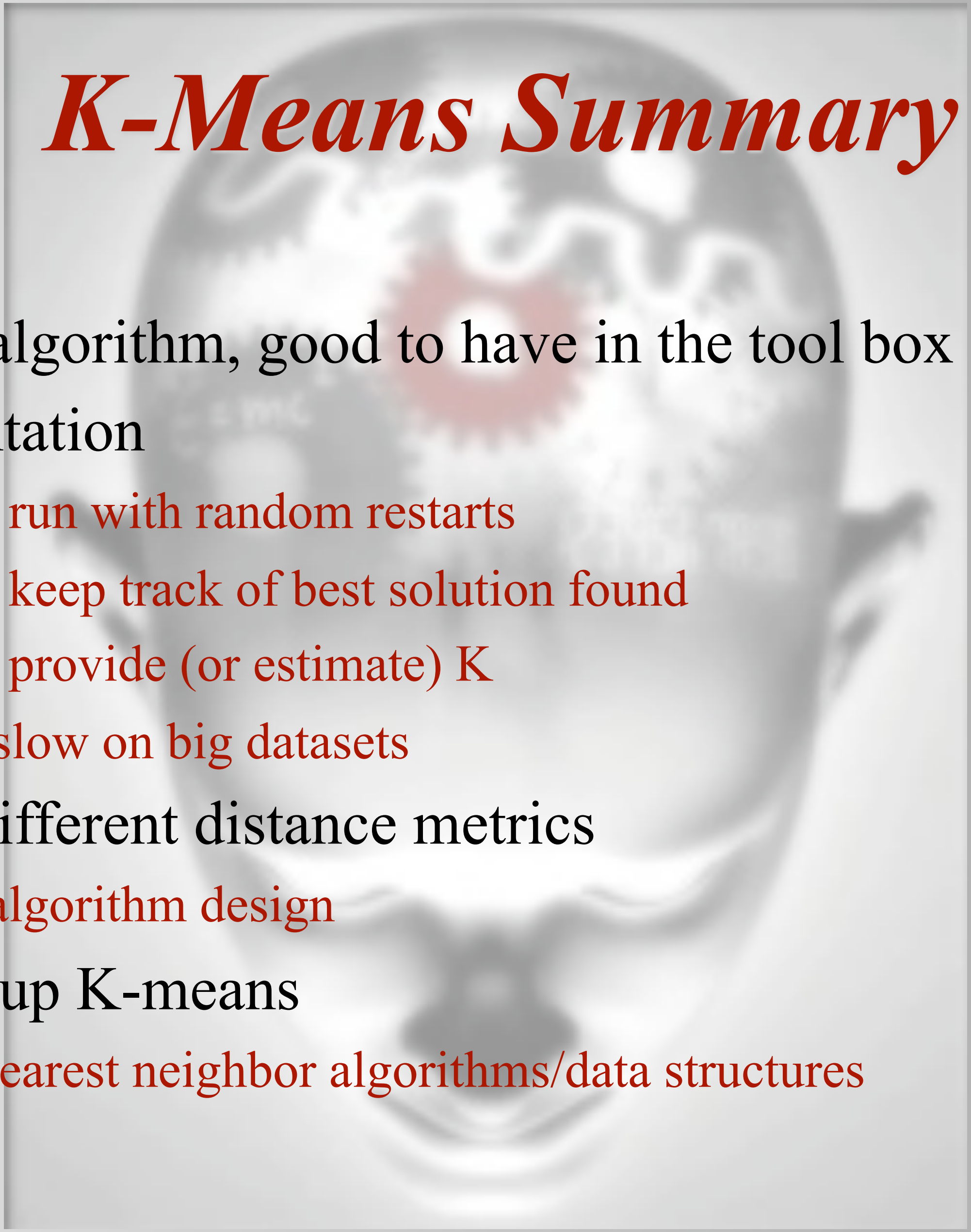
How Many Clusters?



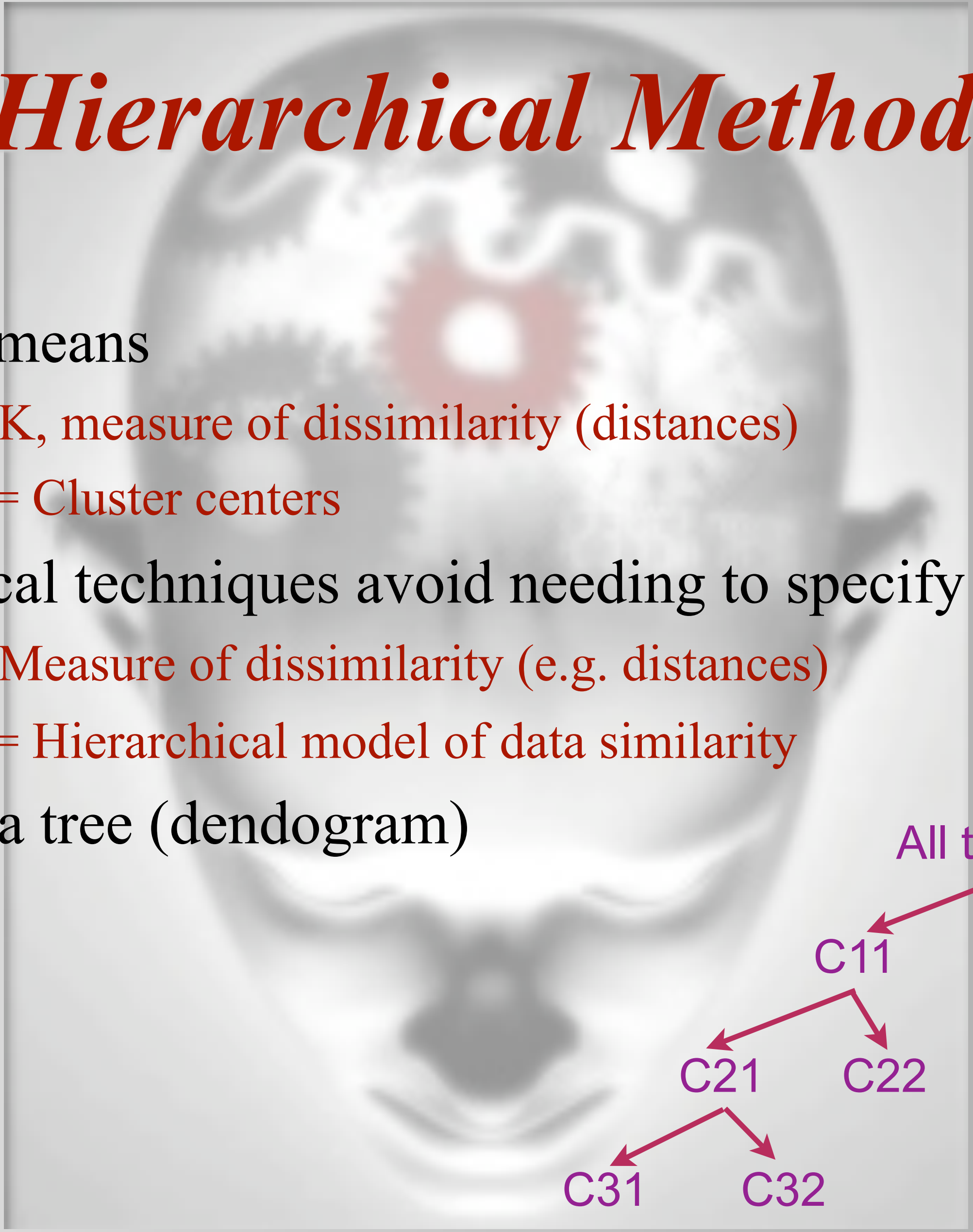


K-Means Summary

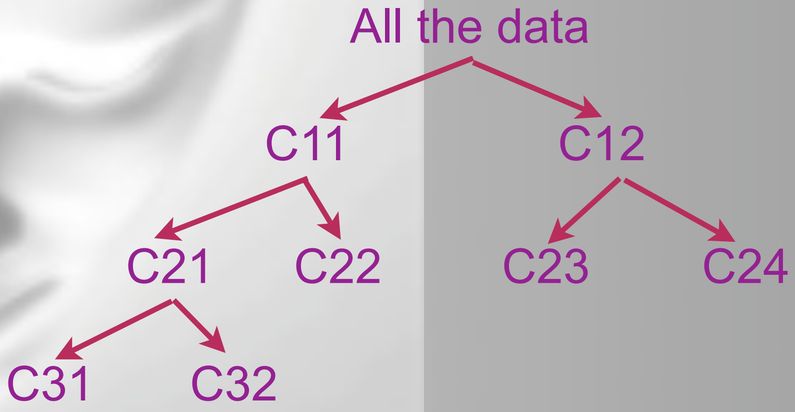
- Practical algorithm, good to have in the tool box
- Implementation
  - Need to run with random restarts
  - Need to keep track of best solution found
  - Need to provide (or estimate) K
  - Can be slow on big datasets
- Can use different distance metrics
  - Part of algorithm design
- Speeding up K-means
  - Faster nearest neighbor algorithms/data structures

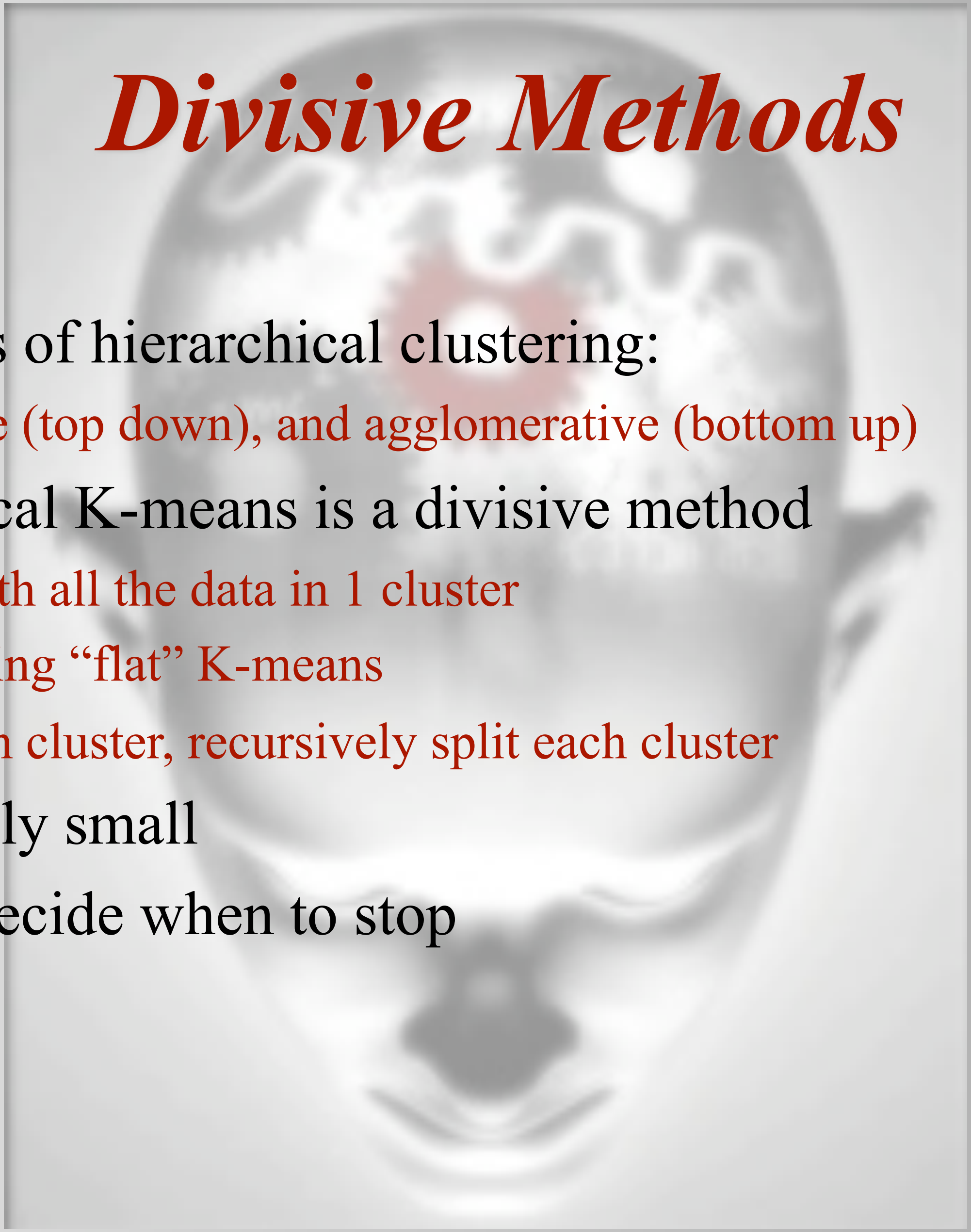


Hierarchical Methods



- Recall K-means
  - Input = K, measure of dissimilarity (distances)
  - Output = Cluster centers
- Hierarchical techniques avoid needing to specify K
  - Input = Measure of dissimilarity (e.g. distances)
  - Output = Hierarchical model of data similarity
- Output is a tree (dendogram)

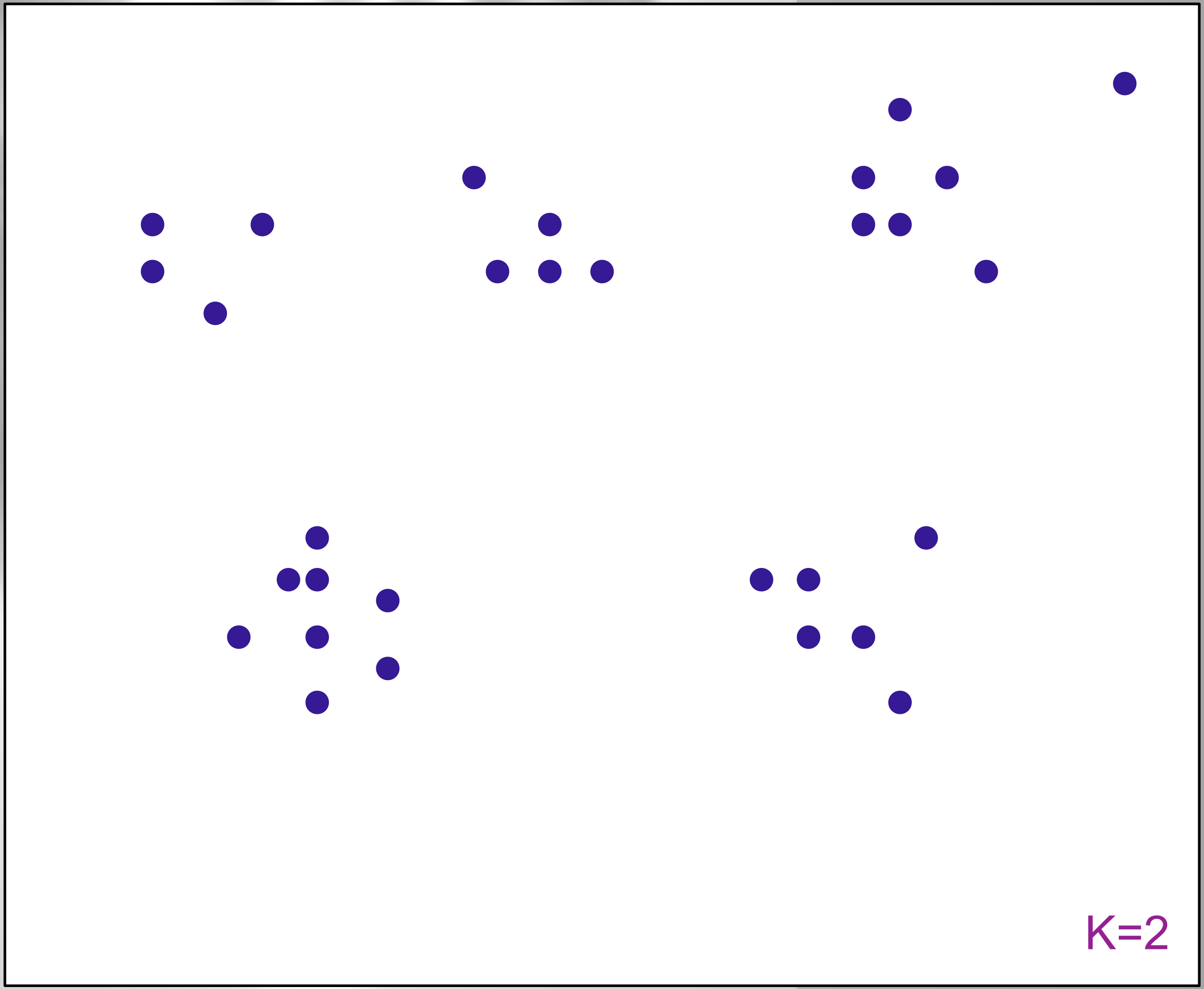




Divisive Methods

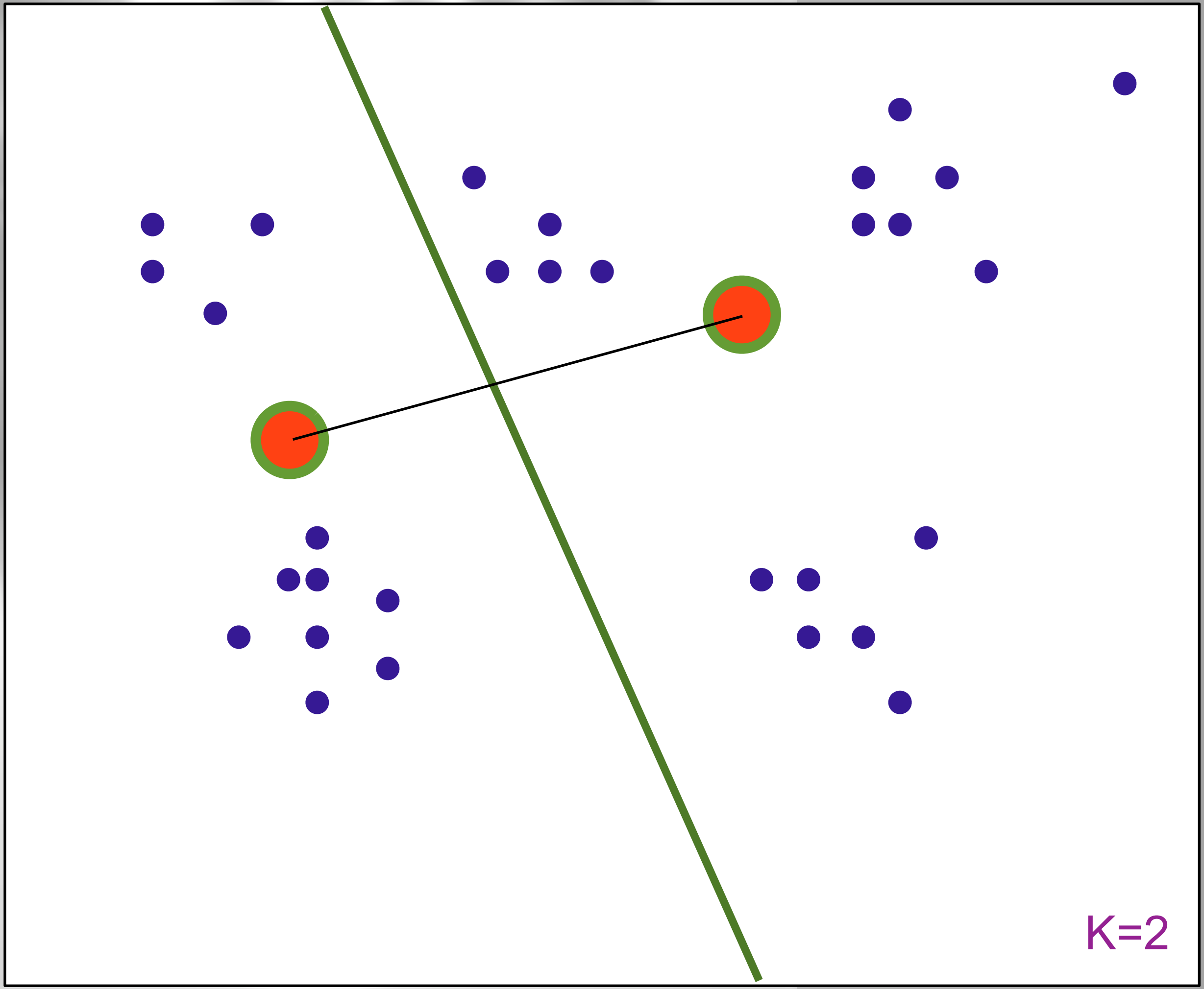
- Two types of hierarchical clustering:
  - Divisive (top down), and agglomerative (bottom up)
- Hierarchical K-means is a divisive method
  - Start with all the data in 1 cluster
  - Split using “flat” K-means
  - For each cluster, recursively split each cluster
- K is usually small
- Need to decide when to stop

Hierarchical K-Means

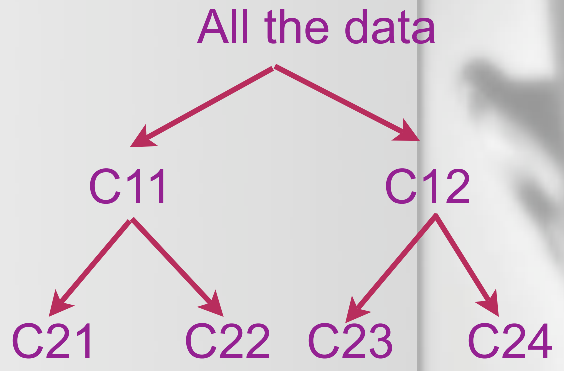


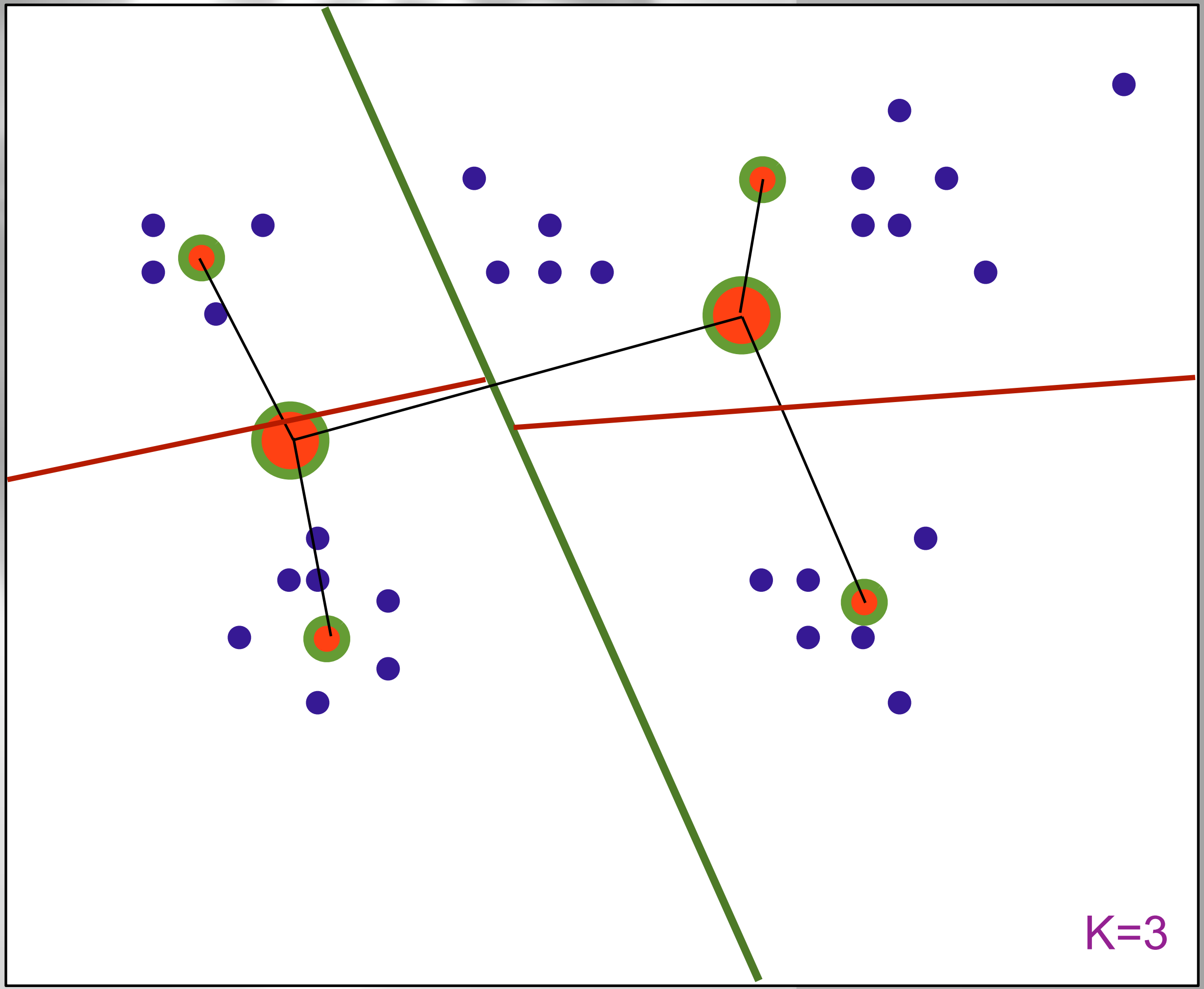
Hierarchical K-Means

$$\begin{tikzcd}
 & \text{All the data} \arrow[dl] \arrow[dr] & \\
\text{C11} & & \text{C12}
\end{tikzcd}$$



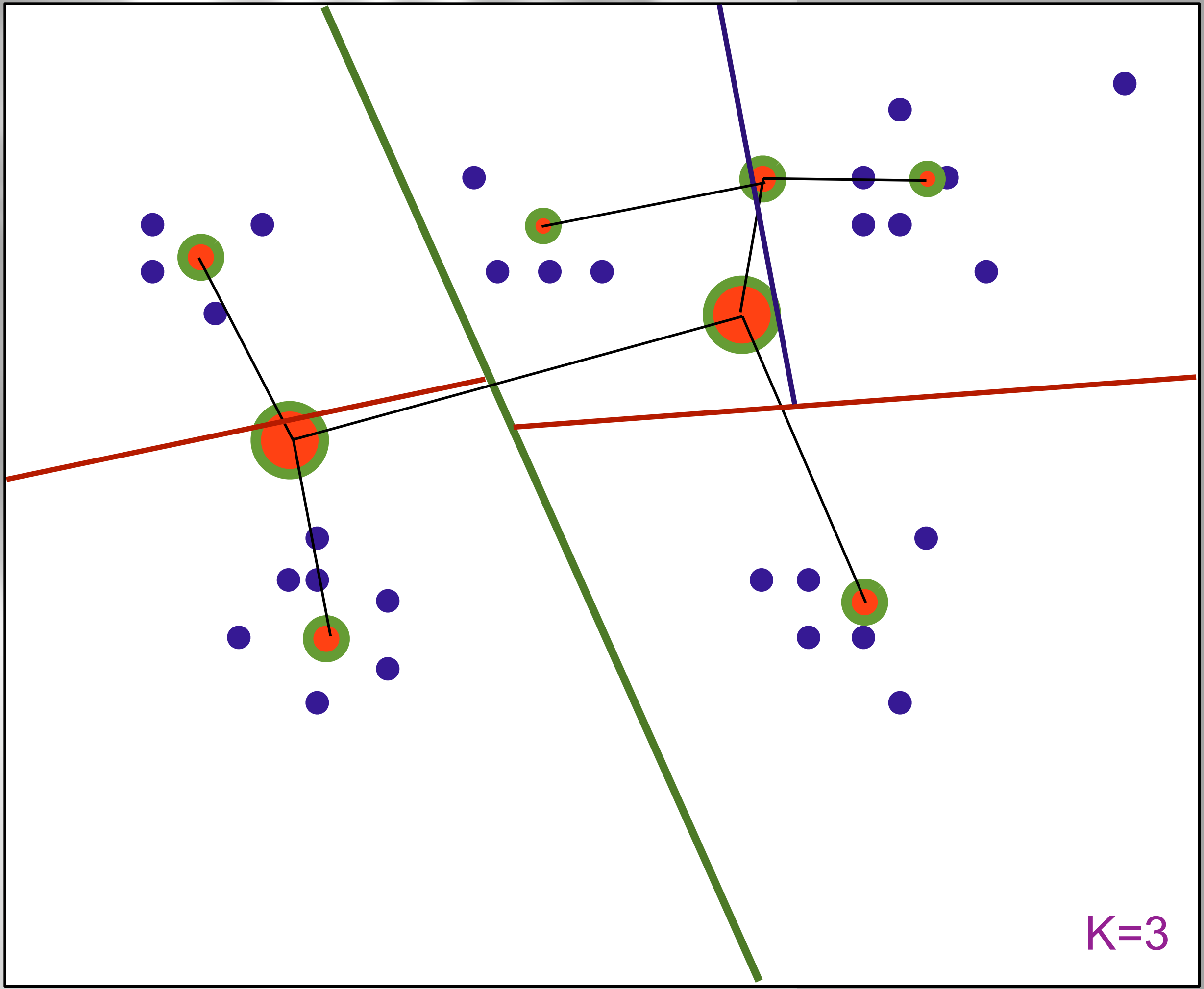
Hierarchical K-Means



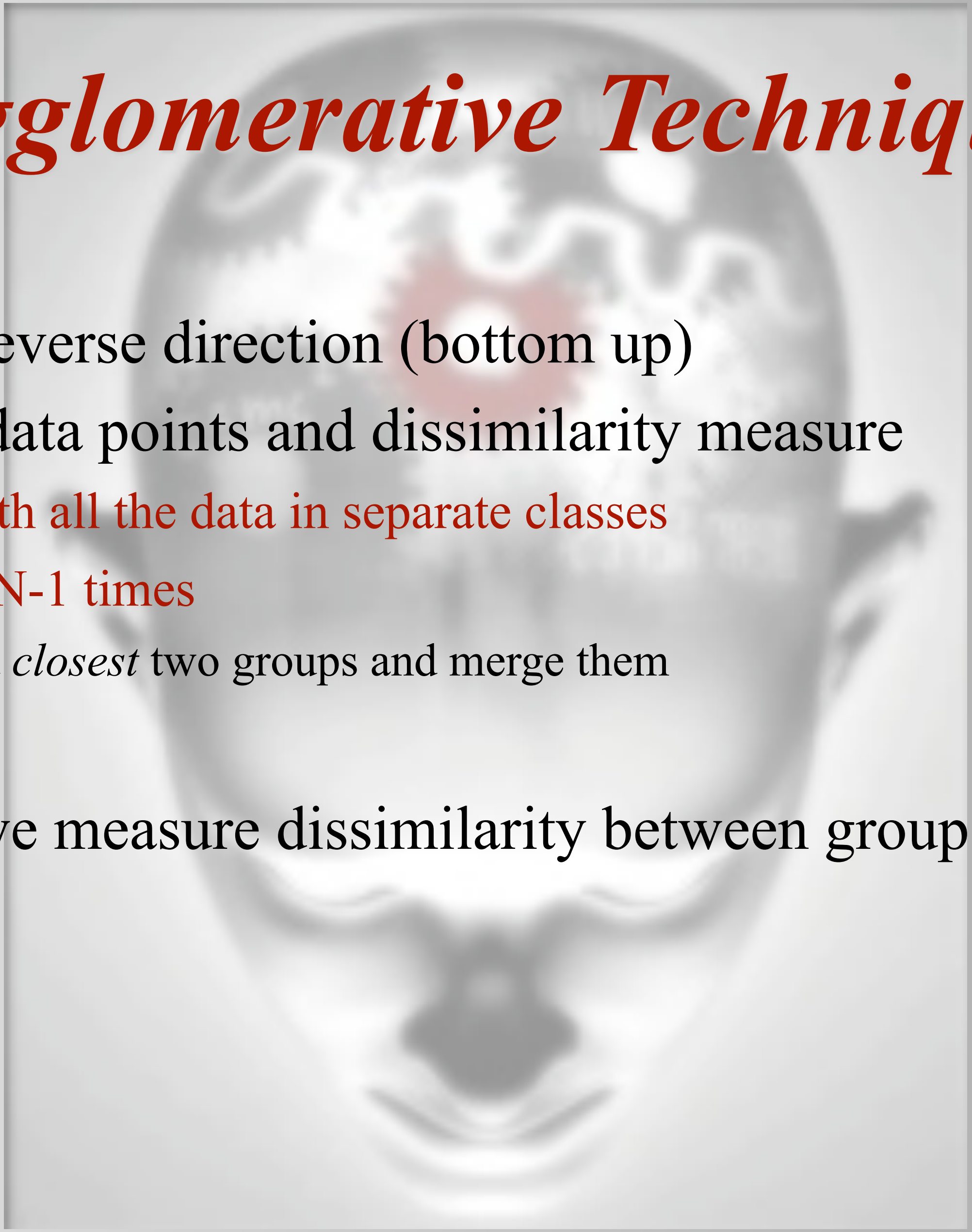


Hierarchical K-Means

$$\begin{tikzcd}
 & & \text{All the data} \arrow[dl] \arrow[drr] & & & & \\
 & C11 \arrow[dl] \arrow[dr] & & & C12 \arrow[dl] \arrow[dr] & & \\
C21 & & C22 & C23 & & C24 \arrow[dl] \arrow[dr] & \\
 & & & & C33 & & C34
\end{tikzcd}$$



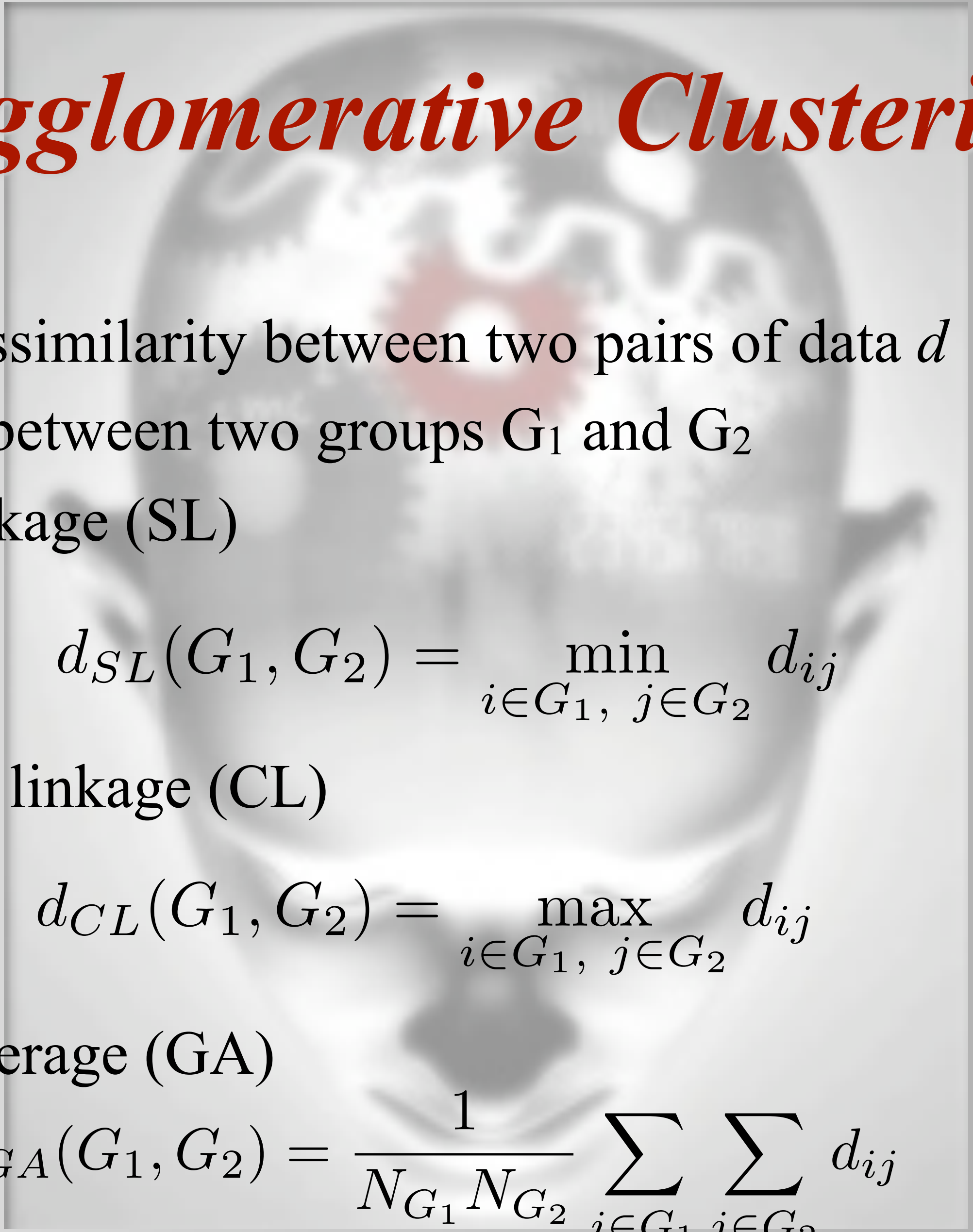
Agglomerative Techniques



- Work in reverse direction (bottom up)
- Given N data points and dissimilarity measure
  - Start with all the data in separate classes
  - Repeat N-1 times
    - Find *closest* two groups and merge them

- How do we measure dissimilarity between groups?

Agglomerative Clustering



- Define dissimilarity between two pairs of data *d*
- Distance between two groups G1 and G2
- Single linkage (SL)

$$d_{SL}(G_1, G_2) = \min_{i \in G_1,\ j \in G_2} d_{ij}$$

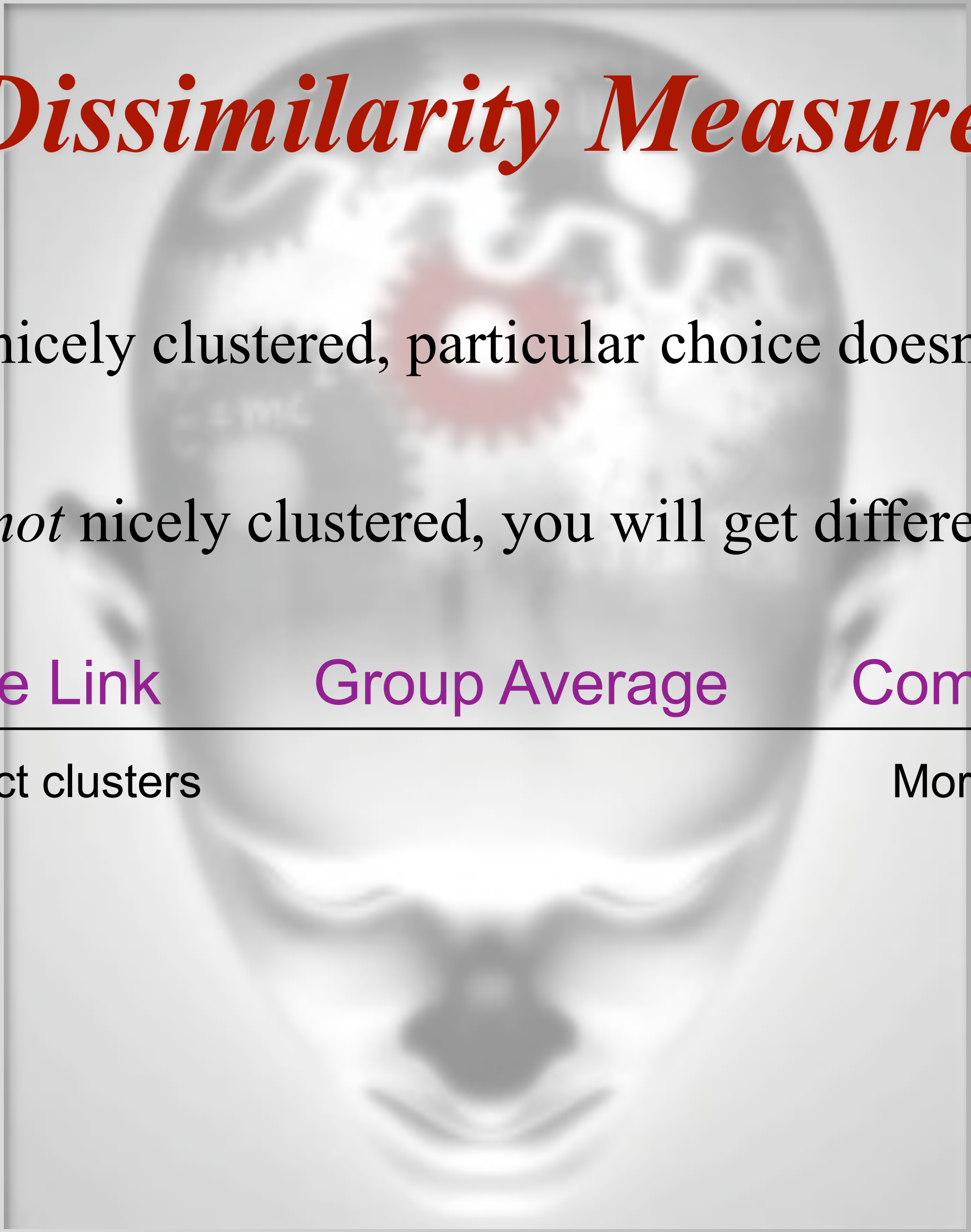
- Complete linkage (CL)

$$d_{CL}(G_1, G_2) = \max_{i \in G_1,\ j \in G_2} d_{ij}$$

- Group Average (GA)

$$d_{GA}(G_1, G_2) = \frac{1}{N_{G_1} N_{G_2}} \sum_{i \in G_1} \sum_{j \in G_2} d_{ij}$$

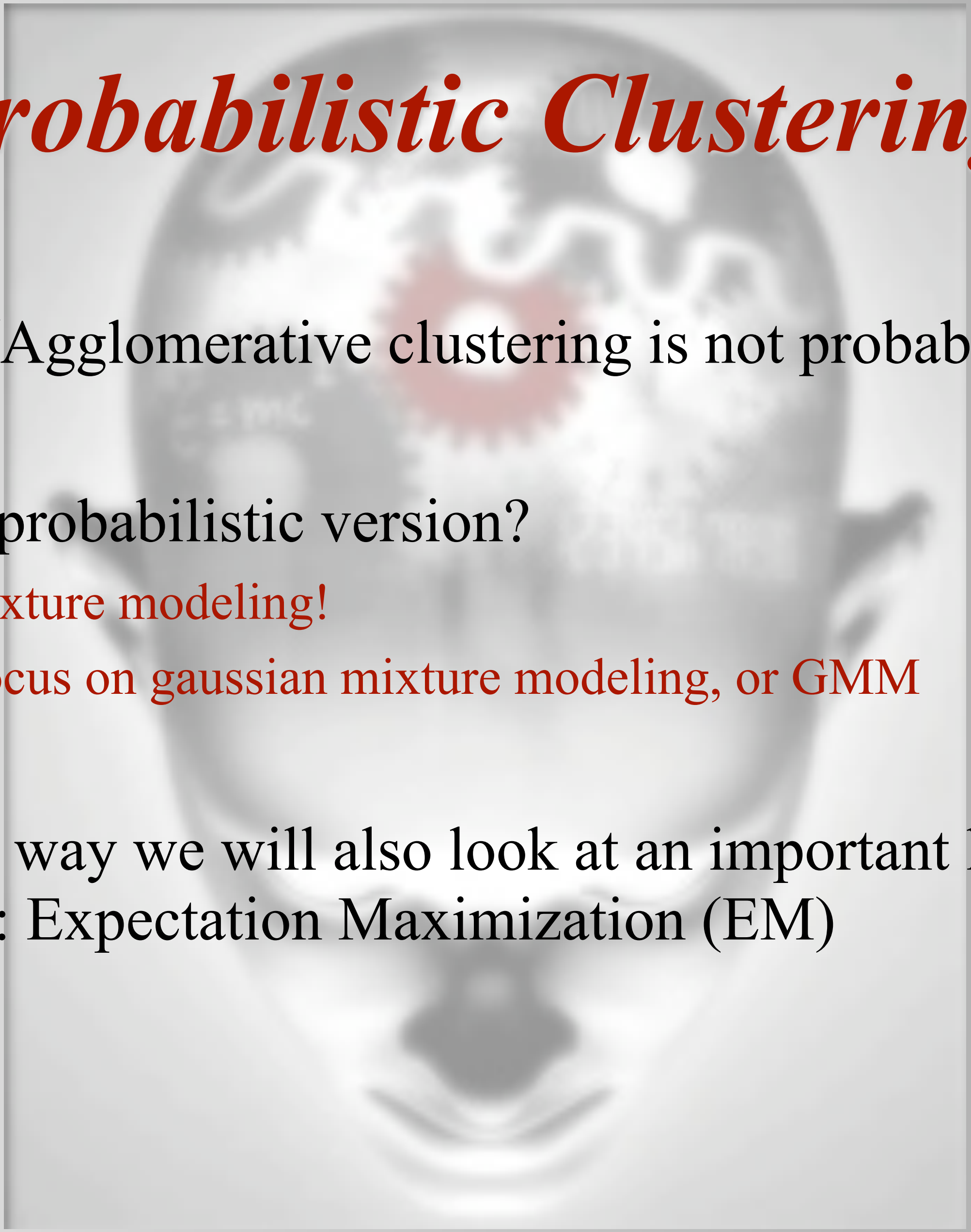
Dissimilarity Measures



- If data is nicely clustered, particular choice doesn't matter
- If data is *not* nicely clustered, you will get different clusters

Single Link ⟵ Group Average ⟶ Complete Link

Less compact clusters (chaining) — More compact clusters

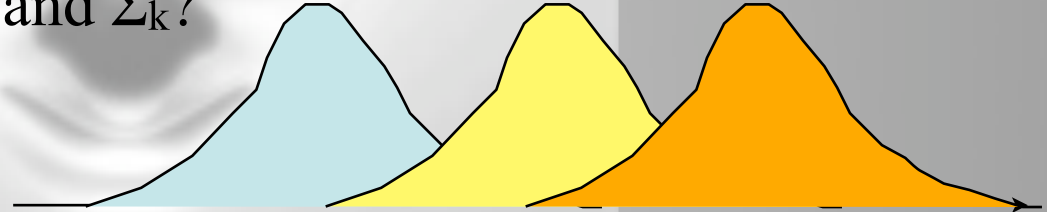


Probabilistic Clustering?

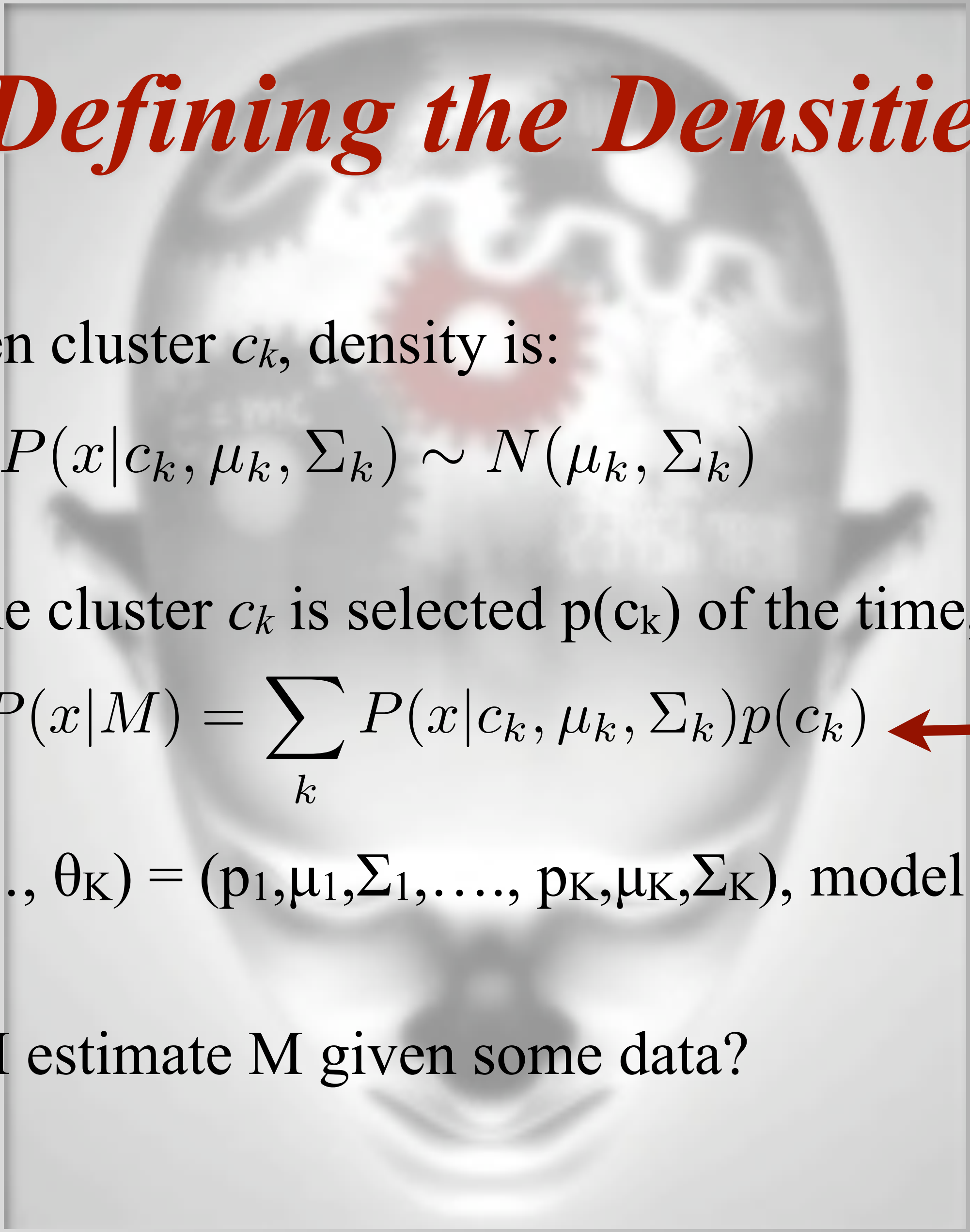
- K-Means/Agglomerative clustering is not probabilistic
- Is there a probabilistic version?
  - Yes! Mixture modeling!
  - We'll focus on gaussian mixture modeling, or GMM
- Along the way we will also look at an important learning technique: Expectation Maximization (EM)

Gaussian Mixture Modeling

- We have some M dimensional data $x_1,\ldots,x_N$
- Let's *assume* that the data is sampled from a set of *K* Gaussian distributions
  - Each Gaussian will correspond to a cluster/class
  - Distribution will be the result of *mixing* the Gaussians
- What is p(x|Model)?
- How can we estimate μ_k and Σ_k?



Defining the Densities



- For a given cluster c_k, density is:

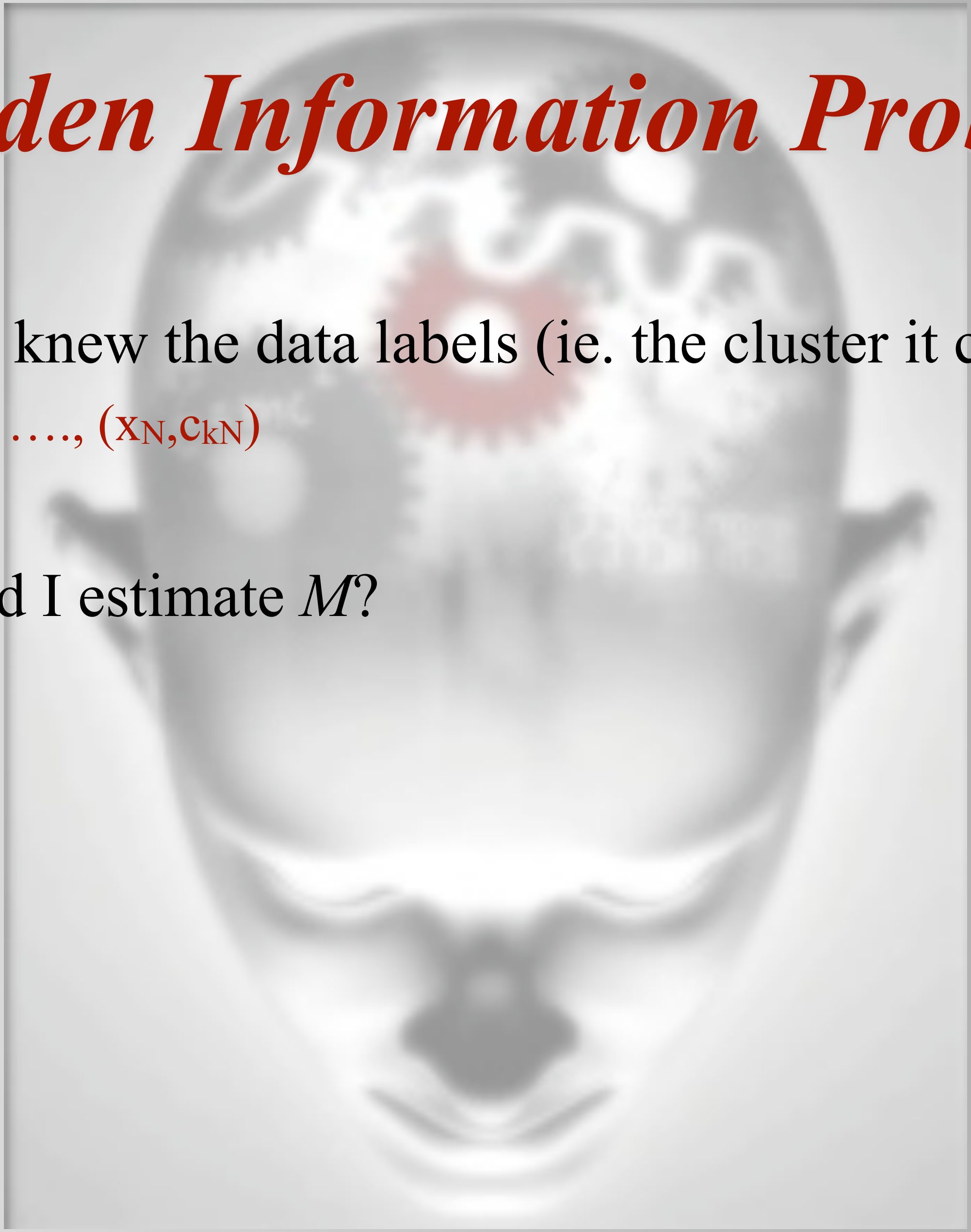
$$P(x|c_k, \mu_k, \Sigma_k) \sim N(\mu_k, \Sigma_k)$$

- We assume cluster c_k is selected $p(c_k)$ of the time, so

$$P(x|M) = \sum_k P(x|c_k, \mu_k, \Sigma_k)\underbrace{p(c_k)}_{\text{Class prior}}$$

- $M = (\theta_1, \ldots, \theta_K) = (p_1, \mu_1, \Sigma_1, \ldots, p_K, \mu_K, \Sigma_K)$, model parameters

- How can I estimate M given some data?

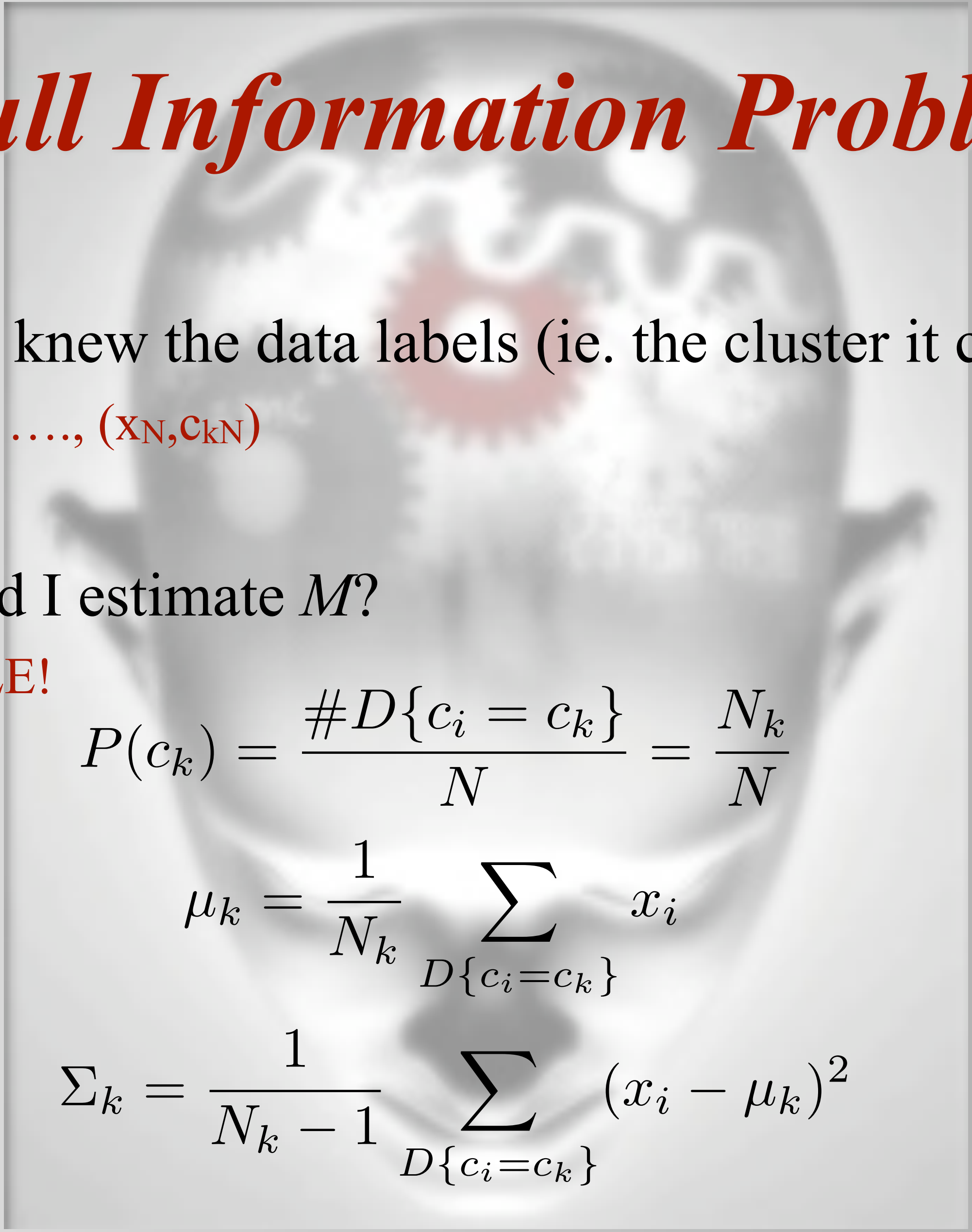


Hidden Information Problem

- Suppose I knew the data labels (ie. the cluster it came from)
  - $(x_1,c_{k1}), \ldots., (x_N,c_{kN})$

- How could I estimate *M*?

Full Information Problem



- Suppose I knew the data labels (ie. the cluster it came from)
  - $(x_1,c_{k1}), \ldots, (x_N,c_{kN})$
- How could I estimate *M*?
  - Use MLE!

$$P(c_k) = \frac{\# D\{c_i = c_k\}}{N} = \frac{N_k}{N}$$

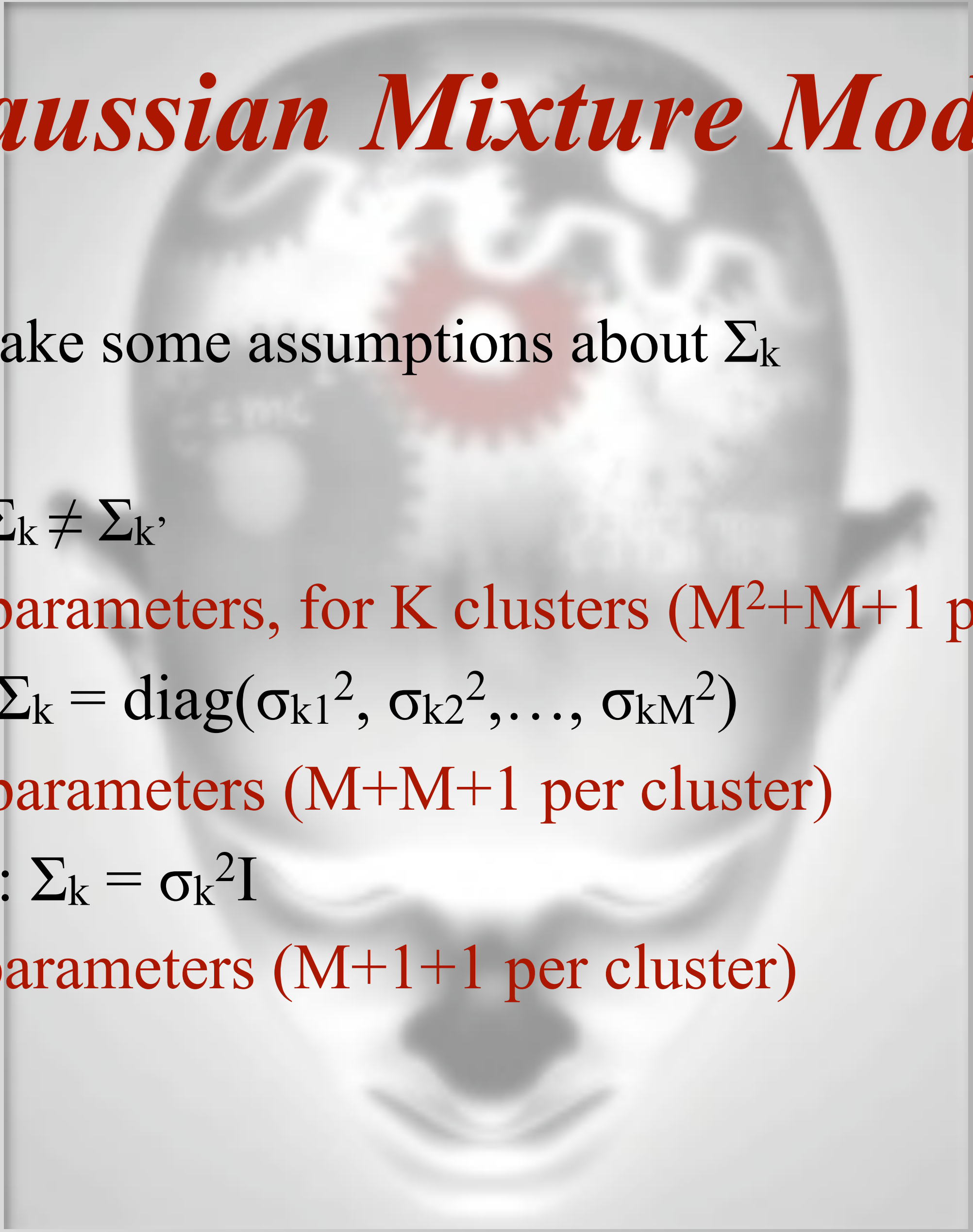
$$\mu_k = \frac{1}{N_k} \sum_{D\{c_i = c_k\}} x_i$$

$$\Sigma_k = \frac{1}{N_k - 1} \sum_{D\{c_i = c_k\}} (x_i - \mu_k)^2$$

Hidden Information Problem

- But we don't know the labels c_k!
  - The labels are *hidden*
- How can we estimate this?
  - Use the EM algorithm!

Gaussian Mixture Models

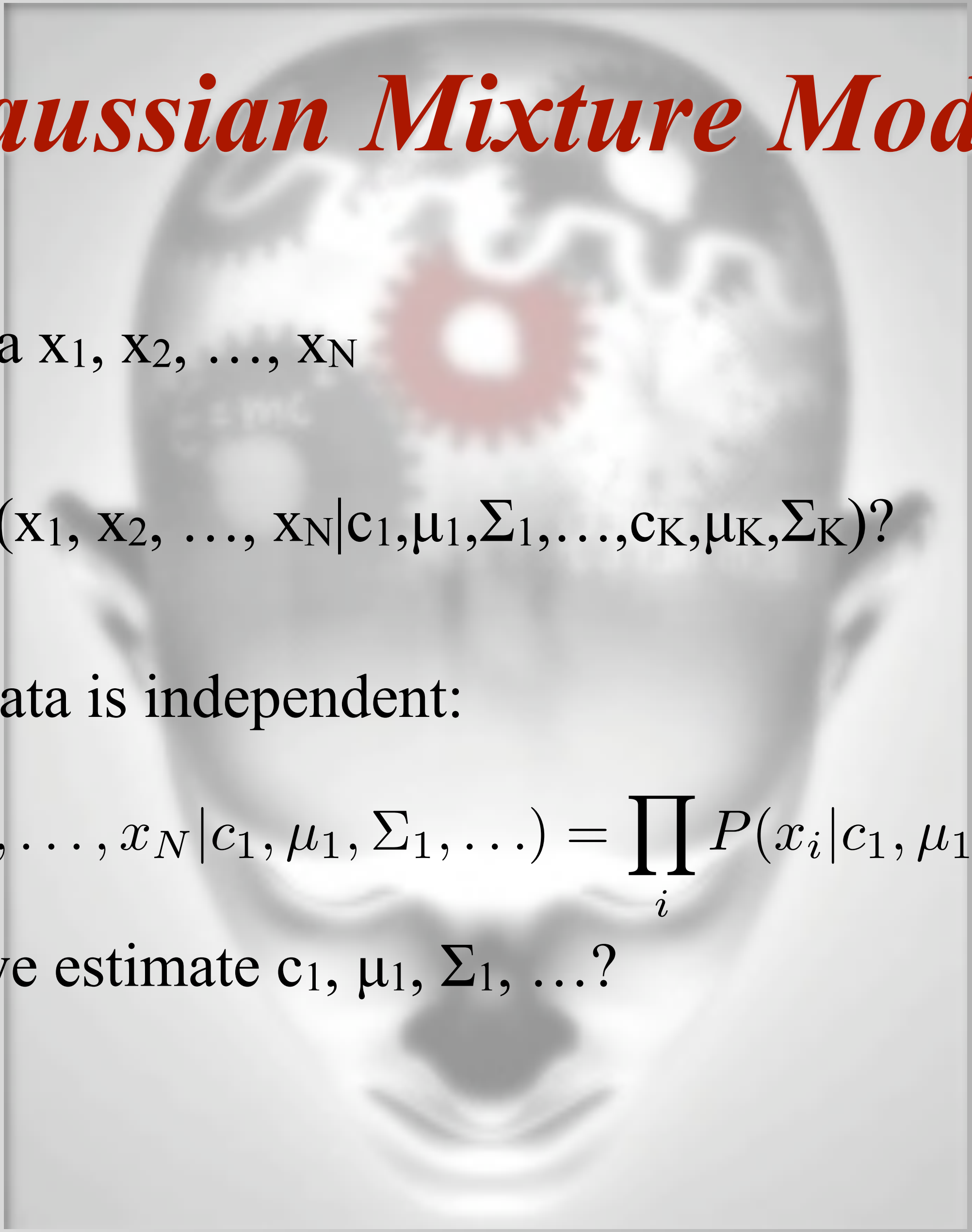


- We can make some assumptions about Σ_k
- General: $\Sigma_k \neq \Sigma_{k'}$
  - $O(K^2)$ parameters, for K clusters (M^2+M+1 per cluster)
- Aligned: $\Sigma_k = \mathrm{diag}(\sigma_{k1}^2, \sigma_{k2}^2, \ldots, \sigma_{kM}^2)$
  - *O(K)* parameters (M+M+1 per cluster)
- Spherical: $\Sigma_k = \sigma_k^2 I$
  - O(1) parameters (M+1+1 per cluster)

Gaussian Mixture Models

- Given data $x_1, x_2, \ldots, x_N$
- What is $P(x_1, x_2, \ldots, x_N | c_1, \mu_1, \Sigma_1, \ldots, c_K, \mu_K, \Sigma_K)$?

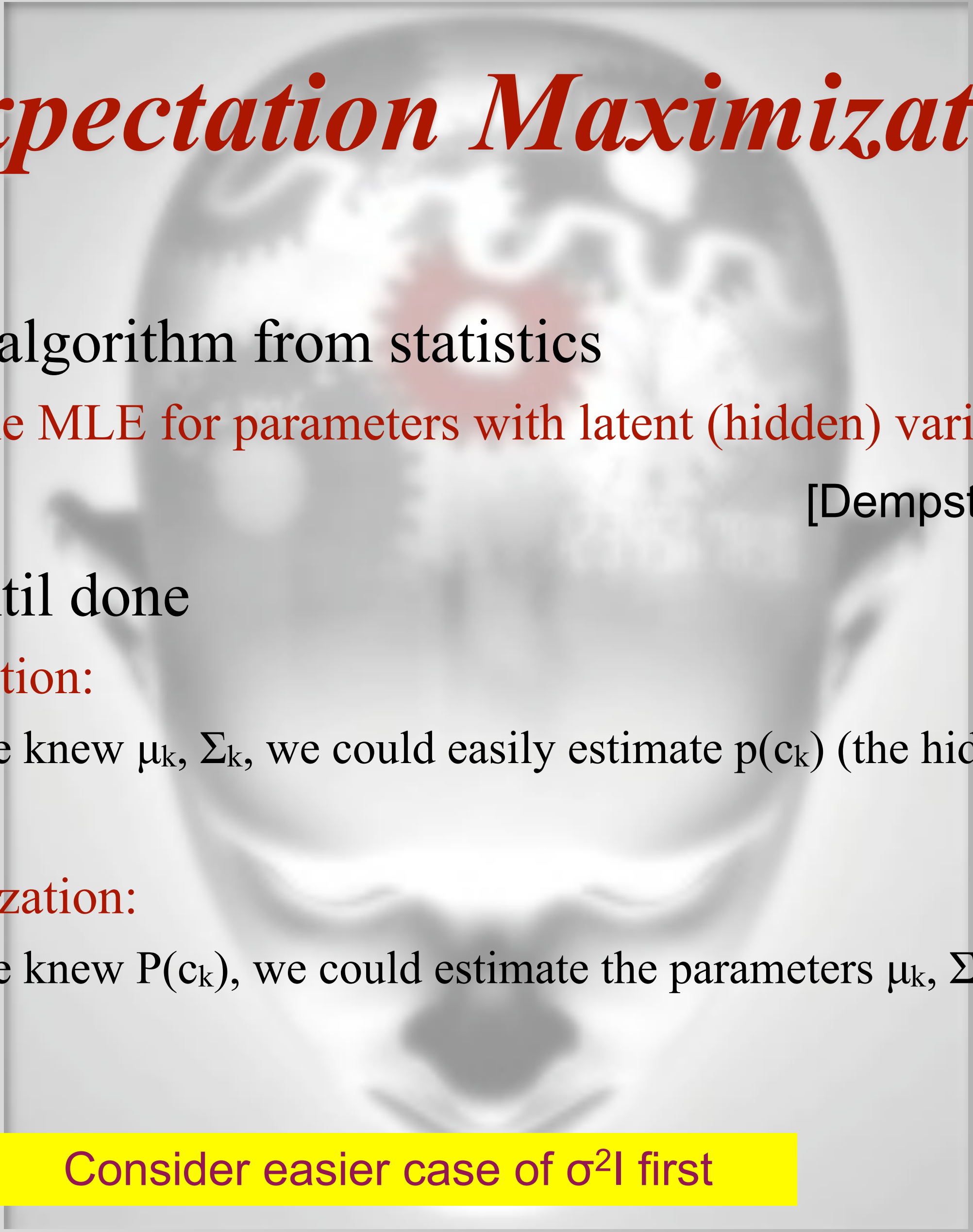
Gaussian Mixture Models



- Given data $x_1, x_2, \ldots, x_N$
- What is $P(x_1, x_2, \ldots, x_N | c_1, \mu_1, \Sigma_1, \ldots, c_K, \mu_K, \Sigma_K)$?
- Assume data is independent:

$$P(x_1, \ldots, x_N | c_1, \mu_1, \Sigma_1, \ldots) = \prod_i P(x_i | c_1, \mu_1, \Sigma_1, \ldots)$$

- How do we estimate $c_1, \mu_1, \Sigma_1, \ldots$?



Expectation Maximization

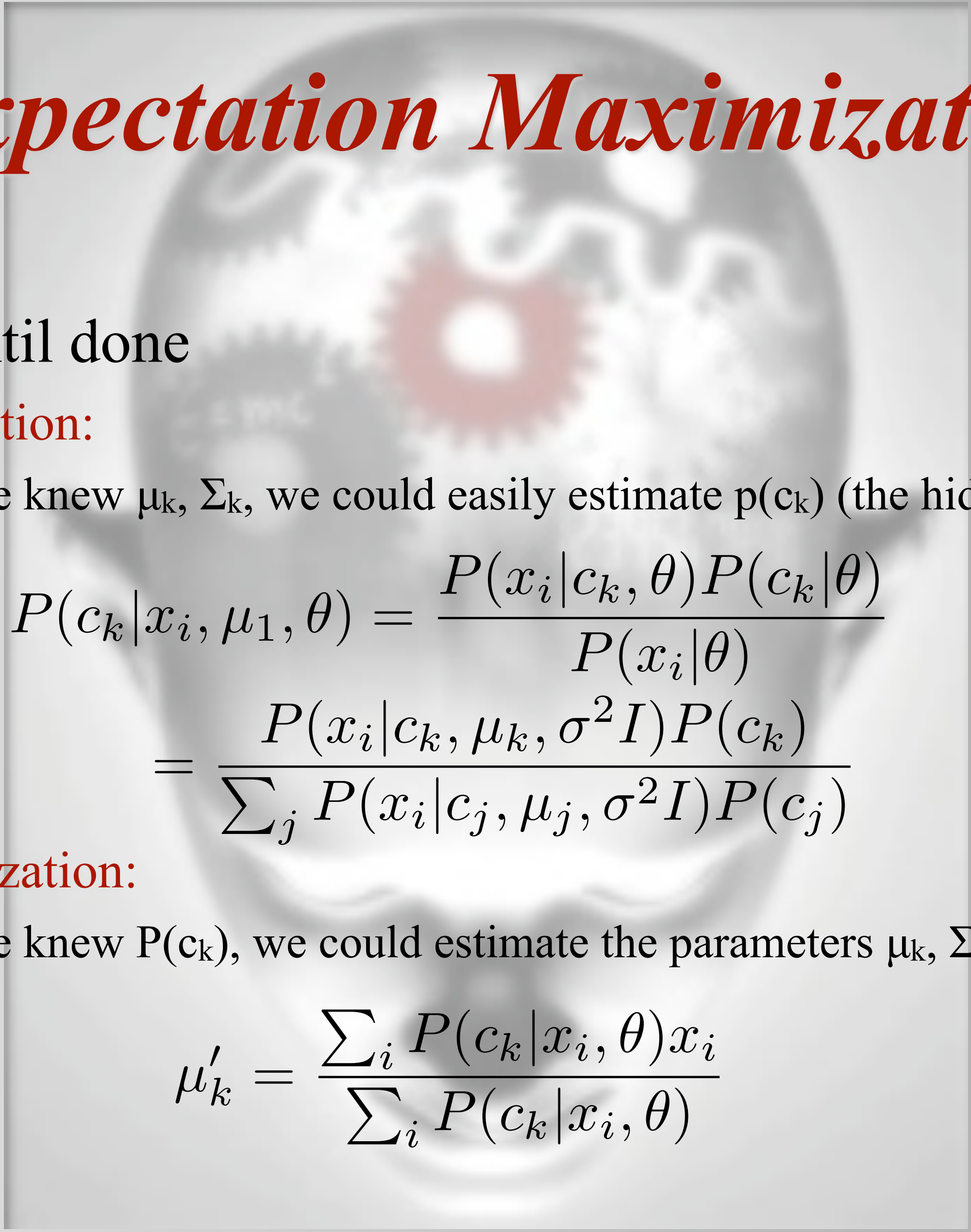
- Powerful algorithm from statistics
  - Finds the MLE for parameters with latent (hidden) variables

[Dempster *et al.* 77]

- Repeat until done
  - Expectation:
    - If we knew μ_k, Σ_k, we could easily estimate $p(c_k)$ (the hidden variables)
  - Maximization:
    - If we knew $P(c_k)$, we could estimate the parameters μ_k, Σ_k

Consider easier case of $\sigma^2 I$ first

Expectation Maximization



- Repeat until done
  - Expectation:
    - If we knew μk, Σk, we could easily estimate p(ck) (the hidden variables)

$$P(c_k|x_i,\mu_1,\theta) = \frac{P(x_i|c_k,\theta)P(c_k|\theta)}{P(x_i|\theta)}$$

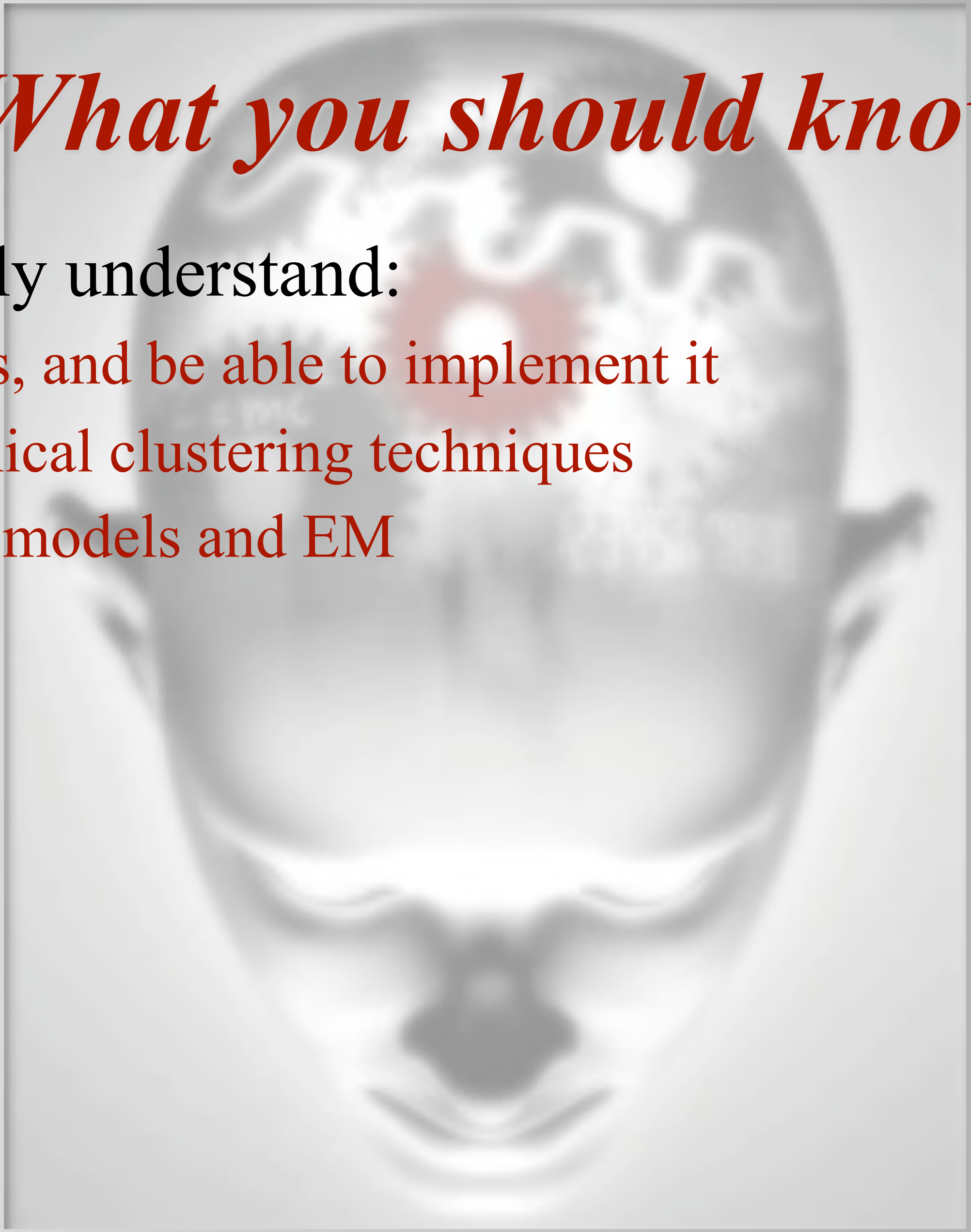
$$= \frac{P(x_i|c_k,\mu_k,\sigma^2 I)P(c_k)}{\sum_j P(x_i|c_j,\mu_j,\sigma^2 I)P(c_j)}$$

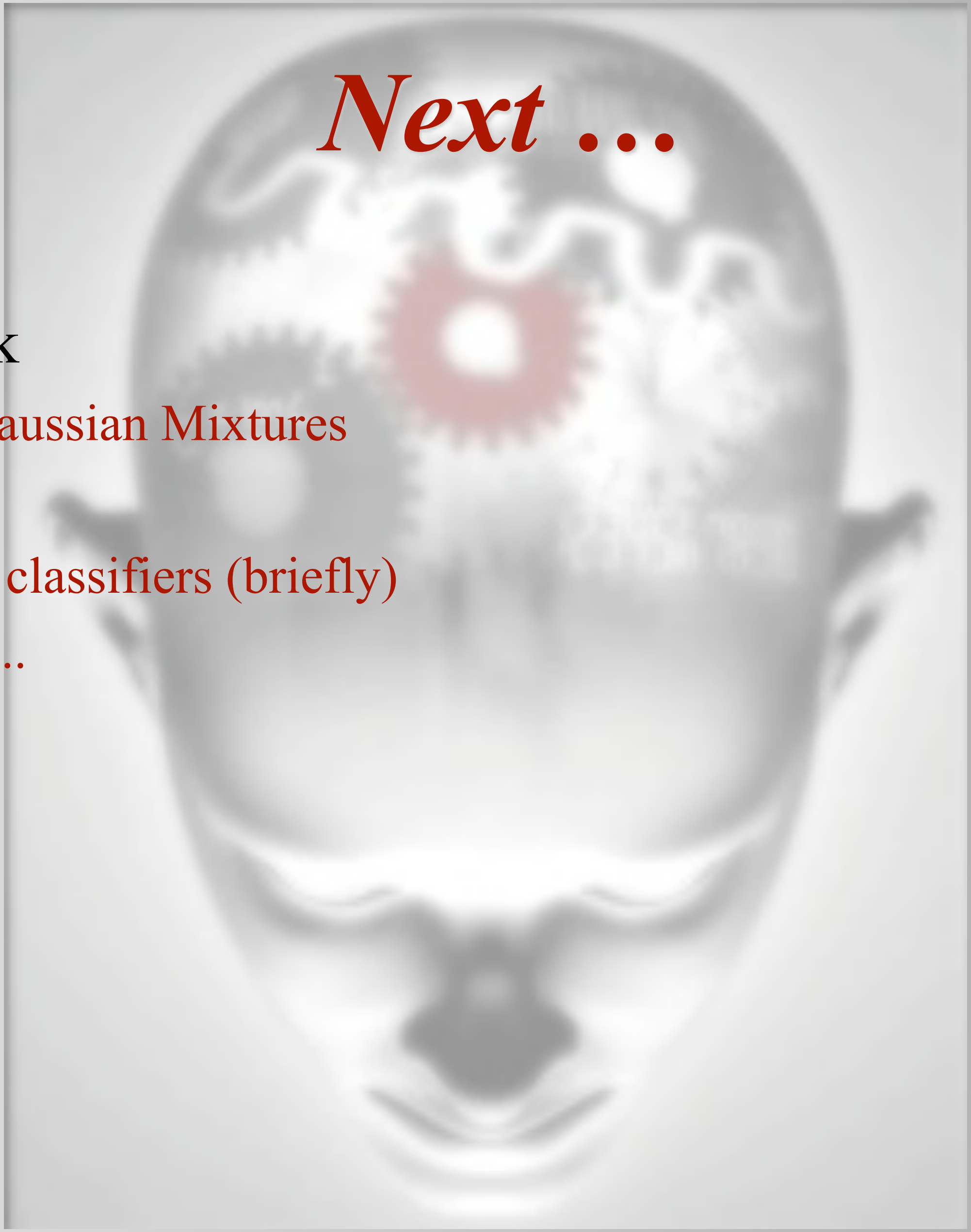
  - Maximization:
    - If we knew P(ck), we could estimate the parameters μk, Σk

$$\mu'_k = \frac{\sum_i P(c_k|x_i,\theta)x_i}{\sum_i P(c_k|x_i,\theta)}$$

What you should know

- Thoroughly understand:
  - K-means, and be able to implement it
  - Hierarchical clustering techniques
  - Mixture models and EM





Next …

- Next week
  - More Gaussian Mixtures
  - PCA
  - Back to classifiers (briefly)
  - Review...